UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

20|21

Trabajo fin de grado

Desarrollo de un sistema de votaciones electrónicas verificables basado en pruebas de conocimiento cero

Alfonso Camblor García

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Desarrollo de un sistema de votaciones electrónicas verificables basado en pruebas de conocimiento cero

Autor: Alfonso Camblor García
Tutor: Alberto Suárez González

junio 2021

Alfonso Camblor García
Desarrollo de un sistema de votaciones electrónicas verificables basado en pruebas de conocimiento cero

Alfonso Camblor García

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mis padres.

The enemy knows the system.

Claude E. Shannon

Prefacio

Este Trabajo Fin de Grado ha sido desarrollado con dos propósitos. El primero, proporcionar una plataforma de voto intuitiva [9], cuyos resultados puedan ser verificados por cualquier usuario [32]. El segundo, el estudio de las técnicas de cifrado que mantienen la anonimidad [19], aspecto fundamental en el desarrollo de votaciones.

Como creador de este documento, deseo que sirva de utilidad para cualquier interesado en la implementación de votaciones electrónicas por Internet. Al mismo tiempo, espero que proporcione una síntesis de las técnicas criptográficas dedicadas a mantener la anonimidad.

Alfonso Camblor García

Agradecimientos

Me gustaría agradecer a la Escuela Politécnica Superior por las herramientas que me ha ido proporcionando a lo largo de los estudios del Grado en Ingeniería Informática.

En particular, quiero destacar el apoyo y orientación brindados por mi tutor, Alberto Suarez, durante todo el desarrollo de este Trabajo Fin de Grado.

También quiero dar las gracias a todas las personas que he conocido a lo largo de estos cuatro últimos años, tanto profesores como compañeros, por sus enseñanzas. Mi agradecimiento a todos ellos.

Un saludo papá y mamá.

Resumen

Las votaciones presenciales traen consigo un importante gasto en organización, propaganda, tiempo, e interferencia en la rutina de votantes y empresas. El desarrollo de un sistema de votaciones electrónicas a través de Internet, permitiría minimizar estos gastos, aportando un medio ágil e intuitivo de participación en el sistema electoral. Asimismo, la crisis sanitaria del COVID-19 sugiere la necesidad de un sistema con estas características.

En este Trabajo Fin de Grado se desarrolla una plataforma de votaciones electrónicas. Esta, garantiza confidencialidad y seguridad mediante las técnicas criptográficas conocidas como pruebas de conocimiento cero. Además, este tipo de sistemas incentivan la participación, pues no requieren de presencialidad, lo que proporciona a los participantes una mayor flexibilidad a la hora de emitir el voto.

La aplicación ha sido desarrollada mediante una arquitectura cliente-servidor, implementada respectivamente con Angular y Django. Gracias a su modularidad, este proyecto queda abierto a futuras iteraciones sobre su funcionalidad, e incluso a la utilización de la existente para dar servicio a otros proyectos.

Palabras clave

Votaciones Electrónicas, Desarrollo Web, Criptografía, Pruebas de conocimiento cero, Identidad digital

Abstract

Face-to-face voting entails a significant expense in organization, propaganda, time, and interference in the routine of voters and companies. The development of an electronic voting system through the Internet would allow these expenses to be minimized, providing an agile and intuitive means of participation in the electoral system. Likewise, the health crisis of COVID-19 suggests the need for a system with these characteristics.

In this Final Degree Project, an electronic voting platform is developed. This guarantees confidentiality and security through cryptographic techniques known as zero knowledge proofs. In addition, these types of systems encourage participation, since they do not require presence, which provides participants with greater flexibility when casting the vote.

The application has been developed using a client-server architecture, implemented respectively with Angular and Django. Thanks to its modularity, this project is open to future iterations on its functionality, and even to the use of the existing one to serve other projects.

Keywords

Electronic voting, Web development, Cryptography, Zero knowledge proofs, Digital identity

Índice

Listas

Lista de algoritmos

Lista de figuras

Lista de tablas

1

Introducción

El avance de las redes de comunicaciones durante las últimas décadas sirve de motivación principal al desarrollo de este Trabajo Fin de Grado. Con el nacimiento de Internet, acaece una digitalización de los medios de comunicación. Entre sus aplicaciones, se proporcionan todo tipo de servicios, desde la transmisión de documentos de texto al desarrollo de sistemas de pago. Estos avances, permiten complementar servicios públicos, como votaciones, con acceso desde Internet, proporcionando un medio seguro y amigable de interacción con el sistema electoral desde la web. Si la votación no requiere de la presencialidad del votante, se conoce como votación remota. Si esta utiliza medios electrónicos en algún punto entre la emisión del voto y su recuento, se considera e-voting, y es el caso que nos concierne.

Las ventajas e inconvenientes de la digitalización de las votaciones es actualmente un debate abierto. Si son presenciales, exigen la participación de un gran número de personas, ya sea como miembros de mesa, interventores o votantes. Por otro lado, en las votaciones remotas, en este caso por Internet, el número de participantes y su trabajo se reducen notablemente. Además, estas últimas incentivan la emisión de voto, ya que sus restricciones pasan a ser únicamente el cumplimiento de la normativa y el acceso a la red. Al mismo tiempo, gracias a la naturaleza mecánica de las votaciones, los ordenadores reducen el coste del proceso, tanto en tiempo como en infraestructura.

Junto a sus potenciales beneficios, las votaciones electrónicas deben cumplir una serie de objetivos que aseguren la integridad de sus resultados. Actualmente y con este fin, el control de participantes y votos es realizado por los integrantes de la mesa electoral, escogidos mediante sorteo, y por los representantes de los candidatos, o interventores. Digitalizar este proceso necesita en algún momento de la identificación de los votantes por parte del sistema. Asimismo, los votos son anónimos, por lo que también se debe garantizar la privacidad de sus emisores. Por último, el proceso debe ser auditable por cualquier agente interesado, a modo de verificación.

1.1. Alcance y objetivos

En las votaciones presenciales, cuando una persona acude a su sede asignada, una vez se comprueba su identidad, y con ella, su derecho a sufragio, se le permite, o no, votar. En el caso de España, el derecho a sufragio se describe en la Constitución [11], y el régimen electoral en la Ley Orgánica 5/1985 [8]. Se expone una explicación del derecho de sufragio activo en [21]. Se deben plantear los requisitos de unas votaciones electrónicas por Internet. Se asume que los votantes se encuentran en un entorno libre de coacción y que los canales de comunicación en la red son seguros extremo a extremo [25]. El objetivo es que cada miembro del censo electoral pueda emitir anónimamente un único voto con una única opción seleccionada. Una vez todos han votado, se recuentan los votos obtenidos por cada opción, y su comparación habilita el anuncio de los resultados. Estos últimos deben ser verificables por cualquier interesado.

La obligatoriedad del voto anónimo se debe a que de lo contrario, se posibilitaría la coacción del usuario al momento de escoger candidatura, e incluso un mercado de compra y venta de votos. Por otro lado, como sólo pueden votar aquellos usuarios con derecho a sufragio, se debe verificar su identidad mediante credenciales. En estas votaciones electrónicas, las máquinas se encargan de demostrar que todas sus fases se han completando según las directrices. Para ello, se le proporciona a todo agente un mecanismo para verificar el proceso. Se parte del documento [9], donde son expuestos diferentes aspectos a tener en cuenta para habilitar sistemas de votaciones electrónicas en la Unión Europea. El primero es la seguridad del ciudadano de que su elección se mantenga en secreto. El segundo, la falta de transparencia en el conteo y almacenamiento de los votos. El tercero, la posibilidad de ataque al sistema encargado de la votación, y por último, la prevención del voto múltiple.

En base a ello, se debe diseñar un sistema de votaciones remotas por Internet que garantice:

1.– **Voto anónimo**: Sólo el emisor del voto debe conocer la candidatura a la que ha votado.
2.– **Legitimidad**: Sólo los usuarios del censo tienen derecho a voto.
3.– **Elegibilidad**: Un votante votará una única vez, y todos sus votos serán genuinos.
4.– **Integridad**: El resultado debe reflejar el verdadero recuento de todos los votos legítimos.
5.– **Robustez**: El sistema debe soportar fallos y prevenir la corrupción.

Para garantizar el voto anónimo, se utilizan firmas criptográficas a modo de clave pública. El diseño basado en estos principios proporciona un mecanismo de validación de voto al que se le pueden añadir complementos, que en este caso son las pruebas de conocimiento cero. Estas, aseguran anonimidad al votante durante la emisión de su voto. Para proveer a la aplicación de tolerancia a fallos, la solución escogida, descrita en el capítulo 3, propone una arquitectura que garantiza la integridad de la información. Los fallos locales no pueden ser suficiente como para detener totalmente el sistema. En el capítulo 2 se detallan los problemas introducidos y sus potenciales soluciones a partir de criptografía. Durante el capítulo 4 se comprueba la consecución de los objetivos propuestos, y el documento cierra con unas conclusiones y propuestas de ampliación, que se detallan en el capítulo 5.

2

Estado del arte

Los sistemas de voto electrónico proporcionan grandes beneficios, ya que facilitan la interacción con el sistema electoral. Además, hacen posibles las votaciones remotas por Internet [32]. Cuando el envío y almacenamiento de los votos se proporciona mediante dispositivos electrónicos, se utiliza el término e-voting. Al mismo tiempo, para cumplir los objetivos, estos sistemas deben resolver ciertos retos técnicos. Los principales son garantizar la anonimidad del voto, evitar el voto múltiple y proporcionar mecanismos de validación y auditoría del proceso.

Cada vez son más los países que incorporan tecnología durante el proceso electoral. En Europa, Bélgica es un país pionero en estas aplicaciones, empezando desde 1989, cuando el método se basaba en bandas magnéticas. A partir de 1994, el marco legal para el voto electrónico fue establecido. Desde ese momento, y a escala global, los sistemas de voto electrónico han ido desarrollándose. Actualmente, la confidencialidad del voto, así como la verificación del seguimiento de las directrices, se logra mediante técnicas criptográficas. Las dificultades encontradas en los sistemas de votación remota actuales residen en la resistencia a la coacción y la identificación de usuarios.

Para demostrar la integridad de estos sistemas, se mide la “Verificabilidad extremo a extremo” [6], que estudia la verificabilidad de cada paso de la votación. Esto permite demostrar la corrección del resultado, asegurando así la integridad de las votaciones. Se considera “Verificabilidad Individual” de un paso si puede ser comprobado por cada miembro del sistema, y “Verificabilidad Universal” cuando la comprobación puede realizarse por cualquier agente externo al mismo. El voto puede ser cambiado o ratificado únicamente antes de su envío, y pasa a ser inmutable una vez almacenado.

2.1. Voto electrónico por Internet

Entre los principales proveedores de voto electrónico en la red, se encuentran los proyectos Helios [2], Polys [20], y ElectionBuddy [15].

Helios es un proyecto Open-Source que implementa un sistema de voto electrónico verificable extremo a extremo, permitiendo comprobar su validez en cada fase. Utiliza además cifrado homomórfico [22], y se descifra únicamente durante el escrutinio final. El usuario recibe también un número de

seguimiento con el que comprobar el estado de su voto. Un aspecto interesante de este sistema es su compatibilidad con la identificación mediante Google. Su característica principal es que, para reducir el riesgo de coacción, permite la emisión de varios votos, de entre los que únicamente el último se tiene en cuenta.

Polys, creado por Kaspersky, es un sistema que utiliza tecnologías de cadenas de bloques. Implícitamente, esta tecnología posibilita la transparencia durante todo el proceso electoral y la inmutabilidad de los votos. Mediante firmas ciegas [7], garantizan la anonimidad de los votantes. Al utilizar una base de datos descentralizada, esta aporta tolerancia a fallos y asegura la integridad de la información.

ElectionBuddy ofrece servicios de votaciones a organizaciones y universidades. Su protocolo utiliza un sistema de claves pública y privada para mantener la lista de usuarios que han votado. Es posible emitir tanto votos públicos como anónimos. La aplicación es auditable, pues se permite la verificación de los votos. Los miembros del censo relativo a una votación que vaya a comenzar, serán comunicados al inicio de esta, incentivando el voto. Además, da mucha flexibilidad a las directrices de las votaciones, permitiendo desde voto único a selección múltiple.

2.1.1. Comparativa

Las tres plataformas cumplen con los requisitos de seguridad de unas votaciones electrónicas remotas. Sus diferencias son principalmente el esquema de cifrado y almacenamiento, que resuelven distintos casos de uso del voto electrónico. Por ello, son objeto de estudio la reducción del impacto de la coacción, el mantenimiento de la integridad de los datos, y la accesibilidad del usuario.

El sistema debe ser resistente a coacción, y dado que los adversarios no pueden verificar la validez de las credenciales, esto puede gestionarse dotando a los votantes de varias credenciales, siendo sólo una de ellas válida. En caso de coacción, el usuario puede emitir un voto utilizando una credencial falsa, que el sistema aceptará, pero cuyo voto no será contabilizado. En otro momento, este podrá emitir su verdadero voto utilizando para ello su credencial válida. Helios implementa una mecánica similar en la que un usuario puede enviar múltiples votos, y únicamente se tiene en cuenta el último. Esta solución se basa en la hipótesis de que un usuario coaccionado, debería poder votar de nuevo cuando se encuentre en un entorno de menor riesgo.

Mantener la integridad de los datos es fundamental durante una votación. Estos demuestran la validez de los resultados, y deben ser inalterables una vez almacenados. Para lograrlo, se propone generar dependencia de los votos ya registrados en todo nuevo voto mediante un esquema de cadena de bloques. Así, si algún voto intenta burlar el sistema, sería detectado al verificarse la firma. El enfoque de Polys comparte este punto de vista, y es por ello que se sostiene sobre una cadena de bloques que registra anónimamente los votos. Gracias a esta, los resultados de la votación permanecen públicos y verificables. Al mismo tiempo, implementa firmas ciegas [7], lo que permite verificación de resultados

públicamente, sin comprometer los datos de los votantes.

Finalmente, la accesibilidad engloba requisitos comunes a todas las aplicaciones informáticas en general. En unas votaciones, el usuario debe identificar fácilmente los elementos a través de los que puede interactuar con la plataforma. ElectionBuddy es la aplicación que más se centra en este aspecto de las mencionadas. En ella, la creación de votaciones y especificación de directrices es notablemente intuitiva.

2.2. Tecnologías de voto

La importancia de garantizar seguridad, el mantenimiento de la integridad de los datos, y la accesibilidad del usuario, son factores fundamentales a tener en cuenta durante el desarrollo de esta aplicación. En la sección 2.1.1 son comparados, y a continuación se presenta la selección de métodos considerados para mantener la seguridad del proceso electoral.

Mientras que Helios utiliza el cifrado de ElGamal [30], Polys se basa en firmas ciegas [20], y ElectionBuddy en claves públicas y privadas con encriptación de 256bits [30]. El enfoque propuesto durante el desarrollo de esta aplicación, se basa en un esquema de claves públicas y privadas con firmas digitales basadas en el cifrado de ElGamal, y propone utilizar pruebas de conocimiento cero para demostrar su validez, manteniendo la privacidad como una implementación de firmas ciegas, pero permitiendo la consulta del estado del voto.

Las firmas digitales pueden interpretarse como claves públicas, y las pruebas de conocimiento cero son capaces de demostrar el conocimiento de la clave secreta sin revelarla. Esto provoca que la votación sea completamente transparente, pues no hay necesidad de ocultar la firma del voto, por lo que para realizar un recuento, basta con comprobar que los votos están firmados, y si lo están, contabilizarlos durante el escrutinio.

Las bases de datos distribuídas, si mantienen cierta redundancia, se convierten en un sistema con alta tolerancia a fallos. Como se ha explorado, hay sistemas de votaciones que utilizan tecnologías de cadenas de bloques para mantener la integridad de la información. En este proyecto se propone una base de datos distribuida, MongoDB, que permite comunicación entre nodos, así como considerar la emisión de votos como una transacción. Por su arquitectura, esta base de datos puede sustituirse por una cadena de bloques en futuras iteraciones, como puede ser alguna parachain de Polkadot [28], asemejándose así a la infraestructura de Polys.

Un sistema de voto electrónico debe ser amigable e intuitiva para el usuario. ElectionBuddy resuelve satisfactoriamente este aspecto, y sirve de inspiración a la hora de diseñar la interfaz de la aplicación. El objetivo es que cualquier usuario sea capaz de participar y/o crear votaciones con los parámetros que necesite, independientemente del grado de familiaridad de este con los medios digitales.

En el apéndice A se proporcionan los detalles técnicos de los esquemas de compartición de secretos. Además, se profundiza en las pruebas de conocimiento cero, comentando la heurística de Shamir y presentando los algoritmos de identificación y firma de Schnorr, que demuestran el conocimiento de la clave secreta del criptosistema ElGamal sin revelarla.

3

Diseño e Implementación

Este sistema de voto electrónico proporciona voto anónimo y verificación de resultados pública. Esto será así siempre y cuando se garantice que el mecanismo de identificación del censo electoral, y el mecanismo de autenticación de sus miembros, son válidos. El sistema requiere también de tolerancia a fallos, por lo que debe ser modular. Además, debe garantizar la integridad de la información, evitando corrupción y aportando un protocolo para poder validar que una firma está registrada, así como el secreto que la valida, sin asociarse al voto emitido. A lo largo de este capítulo, se analiza con una perspectiva más técnica el funcionamiento de unas votaciones electrónicas, y se describe el funcionamiento de un sistema sobre el que se proporciona el servicio de voto electrónico. Con este fin, se definen votación y votaciones electrónicas.

Una **votación** es un evento con tres fases sucesivas en un intervalo de tiempo definido, y cuyo propósito es la toma de decisiones que permita a los integrantes de un grupo medir su opinión conjunta sobre una cuestión de importancia común. Durante su primera fase, se elabora un censo con los datos identificativos de todas las personas que pueden participar. Durante la segunda fase, cada participante emite un voto reflejando su opinión de forma anónima, generalmente seleccionando una de entre varias opciones. Por último, se realiza el recuento del número de votos, y el resultado se utiliza para tomar las decisiones por las que se realizó el evento.

Las tareas de cada uno de los agentes involucrados en el evento se separan en dos grupos principales, los organizadores del evento y los participantes. Los primeros se encargan del cumplimiento de la normativa a la que está sujeta la votación, mientras que los segundos se centran en reflejar su opinión mediante la emisión de votos. Las tareas de los organizadores pueden dividirse en tareas de control de la votación y en tareas de recuento de votos. Las tareas de control son aquellas que protegen a la votación de las acciones de agentes que no sigan las normas, y aseguran a todo agente que las siga, que su voto será válido. Las tareas de recuento de votos o escrutinio consisten en, tras la finalización del evento, contabilizar la cantidad de votos obtenidos por cada opción, dándola por concluida. En sincronía, la realización de las tareas individuales de todos los agentes, de acuerdo a las directrices de la votación, permite considerar válidos los resultados del evento.

Una **votación electrónica** es una votación en la que se utilizan medios electrónicos de voto. Como

la interacción de todos los participantes en algún momento es necesaria para que el resultado de la votación sea válido, se requiere de redes de comunicaciones como infraestructura sobre la que tiene lugar la interacción entre participantes. Se propone una aplicación accesible desde internet mediante navegador web. En este proyecto se presenta el diseño de un sistema que proporciona este tipo de votaciones.

3.1. Descripción

La aplicación a desarrollar tiene como fin proporcionar un servicio de votaciones electrónicas utilizable por cualquier grupo de personas, para lo que se debe diseñar una simulación de las votaciones tradicionales. Como estas últimas requieren de organizadores y participantes para ponerse en marcha, el sistema de la aplicación necesita diferenciar entre esos tipos de usuarios. Al igual que en las votaciones tradicionales, el votante no puede ser identificado mediante su voto, y además, este sistema debe demostrar a los participantes que los resultados son correctos, por lo que se permite a cualquier usuario replicar el recuento si así lo desea.

3.1.1. Metodología

El desarrollo necesita de un planteamiento flexible y tolerante respecto de las necesidades del cliente, por ello, el equipo se reunirá semanalmente, adoptando para ello una metodología ágil de desarrollo, y maximizando la tolerancia a cambios de diseño e implementación. De esta forma se reduce la ambigüedad del proyecto progresivamente hasta obtener un producto que encaje satisfactoriamente en el marco descrito por el cliente.

La reunión con el cliente da comienzo a cada iteración, y tiene como fin identificar los puntos clave del proyecto, para luego analizarlos y dar con un diseño que los recoja todos. En la segunda iteración, se presenta una maqueta con funcionalidad reducida que utilizar como base a la que modificar y aumentar la funcionalidad en las sucesivas iteraciones. Cuando dejen de surgir requisitos, y los existentes sean validados por el cliente, comienza la fase de despliegue, durante la que se prueba exhaustivamente la aplicación, y se finaliza la etapa de desarrollo. Durante la última reunión, se aporta una demostración del funcionamiento del servicio, que una vez validada por el cliente, lleva a desplegarlo, volviéndose accesible mediante exploradores web.

3.1.2. Objetivos

Las fases principales de una votación son la convocatoria, el proceso de voto y el escrutinio. Para implementarlas, es necesario definir a los agentes que utilizan el sistema, de forma que las fases quedan concretadas mediante las interacciones permitidas entre estos. En una primera fase se desarrolla

la funcionalidad necesaria para la creación y consulta de votaciones, así como la jerarquía de usuarios de la aplicación. Seguidamente, se desarrolla el proceso de voto y recuento con arquitectura modular, permitiendo implementar más adelante sus requisitos, así como las principales funciones de cada usuario. Para finalizar, se construyen sobre esa arquitectura los protocolos criptográficos que impiden el doble voto y aseguran su anonimidad. Esto implica que la aplicación deberá soportar identificación de usuarios, creación de votaciones, envío de voto, escrutinio y consulta de resultados.

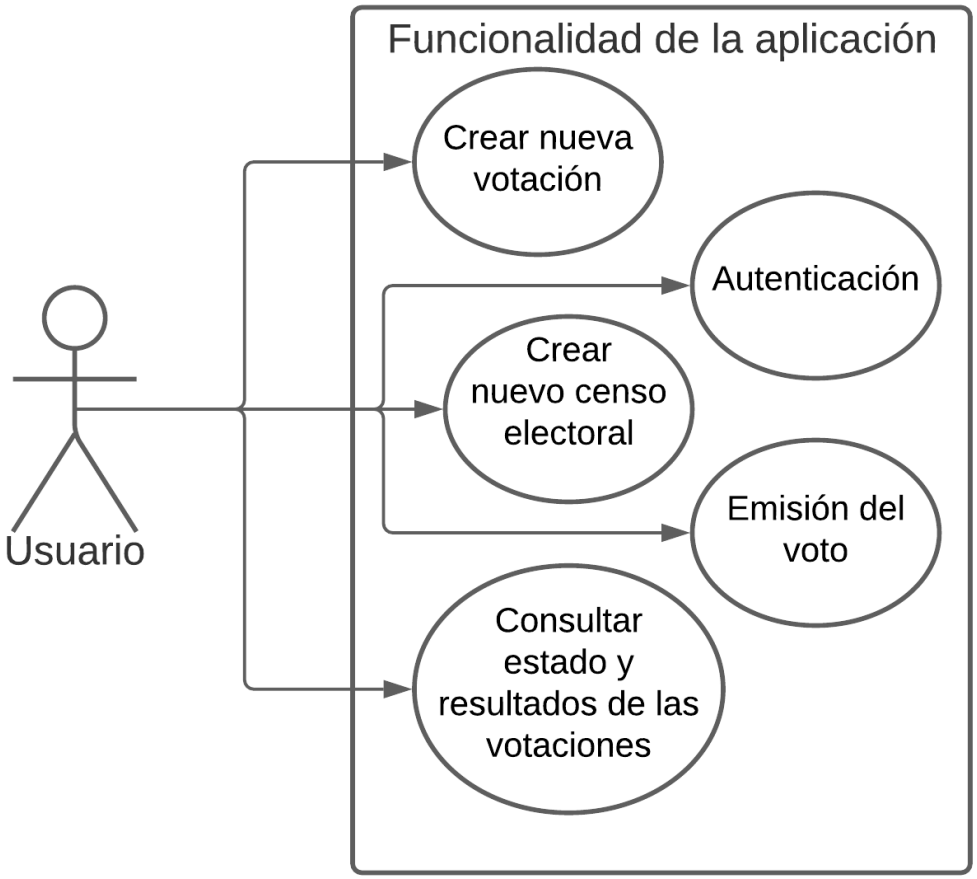

Figura 3.1: Funcionalidad de la aplicación

De todas las fases, el escrutinio de los votos, requerido para la obtención de resultados, es la única en la que sólo interviene el sistema, y comienza automáticamente cuando una votación termina. Las demás funcionalidades son utilizadas en forma de servicios para el usuario de la aplicación, como se ilustra en el diagrama 3.1.

3.2. Requisitos

Un requisito es una especificación técnica de la funcionalidad que la aplicación debe implementar. Dado que durante la fase de desarrollo los objetivos pueden resultar ambiguos, se expresan en una lista que el sistema debe cumplir simultáneamente para proporcionar votaciones electrónicas de uso general. Los requisitos se consideran funcionales si especifican qué debe implementar el sistema de la aplicación para simular unas votaciones, para lo que se definen explícitamente sus fases, concretando así la interacción entre los agente en cada una de ellas. Entre los agentes, se encuentran los encargados del escrutinio de votos y del cumplimiento de la normativa, que pasan a ser sustituidos por el sistema debido a su naturaleza mecánica. Por otro lado, los requisitos se consideran no funcionales cuando proporcionan información sobre cómo deben funcionar los componentes de la aplicación.

Para simular votaciones, se extrae de cada una de sus fases la interacción entre los agentes. Durante la primera, la aplicación debe construir un censo electoral, para lo que necesita implementar la identificación de usuarios y el registro de nuevas candidaturas. La segunda añade funcionalidad para elegir entre candidaturas, así como el envío y almacenamiento de los votos sin registrar la identidad del emisor. Durante la tercera, se recopila el número de votos de cada opción y se obtienen los resultados a partir del recuento.

3.2.1. Requisitos Funcionales

Consisten en la funcionalidad que compone los servicios ofrecidos por la aplicación. Estos son recopilados en la tabla 3.1.

Para proporcionar votaciones, el sistema de la aplicación deberá definir e implementar mecanismos para añadir nuevas y presentar candidaturas a estas, ya que de lo contrario no habría elección en los votos. Asimismo, dado que en las votaciones pueden emitir voto únicamente los agentes inscritos en el censo electoral y estos votos deben ser guardados para realizar el escrutinio al final de la votación, la aplicación debe hacerse cargo de la identificación de usuarios y del recuento de votos.

Entre los agentes que utilizan el sistema hay tres tipos: el primero es el votante y tiene por tarea emitir su voto. Le siguen los creadores, encargados de añadir votaciones y candidaturas. Por último, están los administradores, cuya misión es validar a nuevos creadores. Para implementar estos roles, es necesario diferenciar al usuario de la aplicación, por lo que se establece que todo votante se debe registrar e identificar en el sistema. Dependiendo del tipo de usuario del que se trate, la aplicación ofrecerá distinta funcionalidad.

Requisito
1. Votaciones
1.1. Registro de votaciones
1.2. Registro de candidaturas
1.3. Asignación del censo electoral
1.4. Envío de votos
1.5. Guardado de votos
1.6. Recuento de votos
2. Usuarios
2.1. Registro de usuarios
2.2. Identificación de usuarios
2.3. Distintos tipos de usuarios
2.4. Registro de grupos de usuarios
2.5. Unión a grupos de usuarios

Tabla 3.1: Requisitos funcionales

3.2.2. Requisitos No Funcionales

Engloban este ámbito los requisitos del sistema que no son funcionales. Estos son a su vez clasificados a partir de su impacto en la aplicación en términos de seguridad, usabilidad y accesibilidad. Estos se ilustran en la tabla 3.2.

La seguridad de las votaciones es necesaria para garantizar la legitimidad de los resultados. Por ello, los votos deben permanecer inalterables una vez han sido emitidos, y deben existir dos posibilidades una vez se emiten. La primera de ellas es que el voto no sea validado, razón por la que no se almacena en el sistema, mientras que la segunda corresponde a los votos validados, que son almacenados en el sistema sin ningún tipo de información que permita identificar su emisor. El criterio que se sigue a la hora de validar los votos, es identificar si el usuario forma parte del censo electoral, así como comprobar que el usuario emite un único voto. Teniendo en cuenta lo anterior, es posible implementar votaciones electrónicas resistentes al fraude. Como último apartado, el sistema debe ser tolerante a fallos, es decir, en caso de ocurrir un problema a nivel local, el funcionamiento global del sistema debe permanecer inalterado.

Al tratarse de un sistema de votaciones de uso general, se requiere del cuidado de la usabilidad de la aplicación, facilitando a todo tipo de usuarios un sistema claro y sin ambigüedades. Es por esto que se necesita de un mecanismo para navegar entre todas las secciones del servicio, así como de indicaciones para el usuario sobre su localización en el proceso. Además, los colores, textos e imágenes escogidos deben mantener coherencia, para lo que todos los colores deben pertenecer a una misma paleta, y los textos e imágenes deben estar relacionados, evitando la multiplicidad de

cualquiera de ellos y estandarizando el uso de imágenes y textos con las acciones que el usuario pueda realizar durante la votación. Para garantizar el acceso a la aplicación, se escogen aquellos navegadores web con mayor volumen de usuarios y se asegura la compatibilidad, por este criterio, con Google Chrome, Safari y Microsoft Edge.

Requisito
0. Total
1. Seguridad
1.1. Cada emisión de voto debe ser inalterable
1.2. Cada voto es una transacción, o se almacena o se descarta
1.3. Sólo pueden participar los usuarios inscritos en el censo electoral
1.4. Mantener anónimo al emisor de cada voto
1.5. Prevenir el voto múltiple
1.6. Permitir verificación de resultados
1.7. Los fallos locales no deben afectar al funcionamiento global
1.8. En caso de autenticación fallida, el agente no debe aprender datos del mensaje informativo
1.9. Acceso a servicios dependiendo del tipo de usuario
1.10. Pertenecer a un grupo no proporciona la identidad del votante
2. Usabilidad
2.1. Utilizar barra de acceso lateral para navegación
2.2. Indicar al usuario en que sección de la aplicación se encuentra en todo momento
2.3. Indicar al usuario las acciones posibles en todo momento
2.4. Utilizar colores pertenecientes a una misma paleta
2.5. Reducir la cantidad de información e imágenes
2.6. Identificar acciones mediante imágenes de forma intuitiva
2.7. La lista de votaciones debe presentarse en la pantalla principal
2.8. La consulta de votaciones debe realizarse al pulsar sobre una votación
2.9. La consulta de votaciones debe proporcionar los resultados si la votación ya ha finalizado
3. Accesibilidad
3.1. Compatibilidad con Google Chrome
3.2. Compatibilidad con Safari
3.3. Compatibilidad con Microsoft Edge

Tabla 3.2: Requisitos no funcionales

3.2.3. Gestión del tiempo

Este proyecto ha comprendido un total de 24 semanas. La organización de tareas se indica en la figura 3.2.

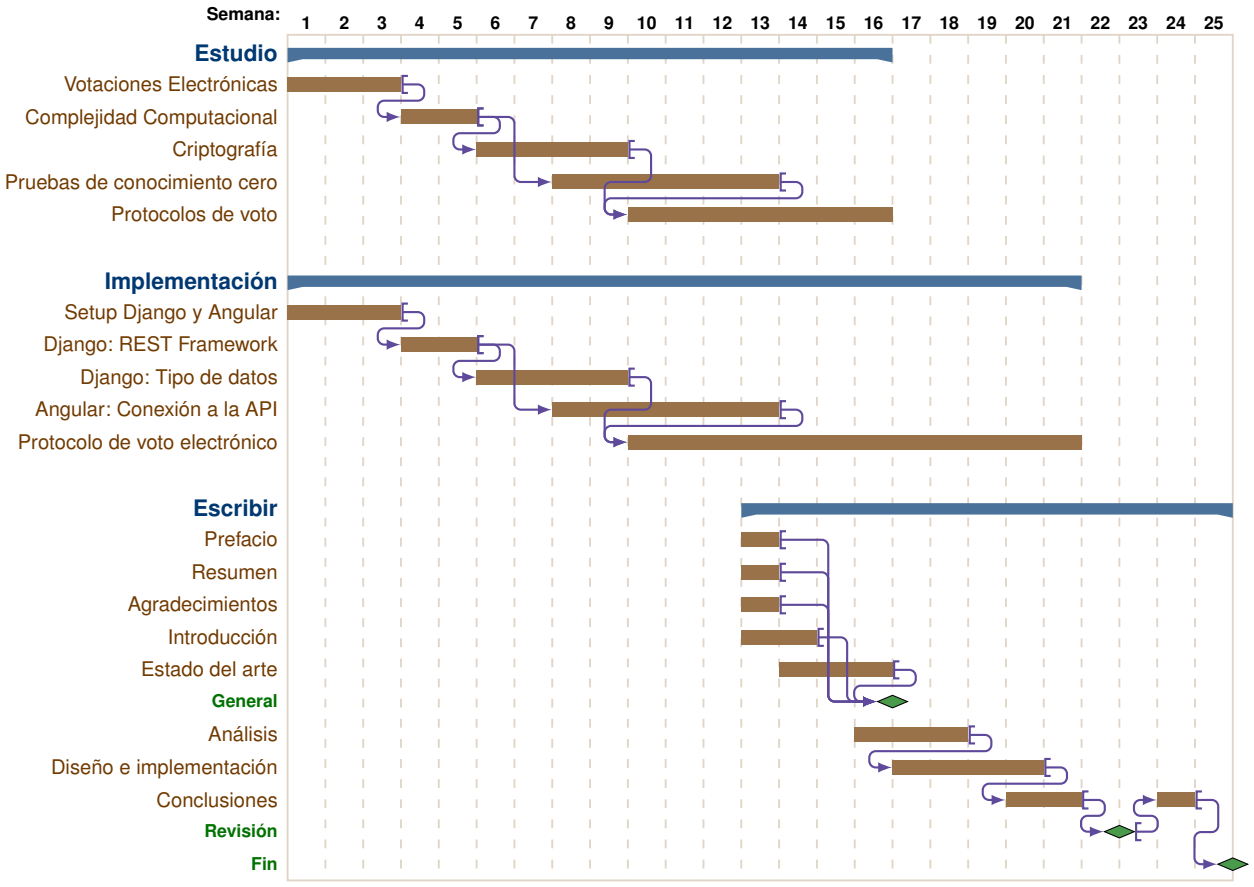

Figura 3.2: Diagrama de Gantt del proyecto.

3.3. Casos de uso

Cada usuario tiene distintas interacciones con el servicio de voto electrónico, pero existen varias comunes a todos ellos. Estas deben definirse de forma concreta para eliminar cualquier tipo de ambigüedad. Para ello, se incluyen sus especificaciones mediante diagramas en Lenguaje unificado de modelado (UML) .

En una votación, sin necesidad de identificación de usuario, se debe tener acceso a cierta información de las votaciones, que consiste en la consulta su estado, y en caso de haber finalizado, de sus resultados, así como la posibilidad de identificarse con el fin de participar activamente en el evento aquellos agentes con derecho a voto. Una vez el usuario ha sido identificado, su tipo caerá dentro de los administradores del sistema, los creadores de votaciones y censos, o los votantes que participan. Estos últimos, interactúan mediante el envío de su voto, que el sistema debe almacenar de forma anónima para realizar el escrutinio al final de la votación. Por otro lado, los administradores y creadores forman parte de los agentes que organizan la votación. Los creadores tienen por tarea añadir nuevas votaciones, censos y candidaturas al sistema. Los administradores, la validación de nuevos usuarios creadores.

3.3.1. Comunes

Al entrar por primera vez en la aplicación, ningún usuario está identificado, pero debe tener acceso a la lista de votaciones y a la consulta de sus estados, donde se incluyen los resultados si la votación ya ha finalizado. Con el fin de agilizar el proceso, la lista se presenta en la pantalla principal, y una vez encontrada la votación pertinente, se consulta pulsando sobre ella. Por otro lado, la identificación tiene su propia sección en la que se introducen el nombre de usuario y la contraseña, se pulsa el botón de acceso y el sistema comprueba la información transmitida. Queda recogidos en la figura 3.3.

Visualizar y consultar detalles de votación

Los detalles disponibles mediante consulta son el estado de la votación, que indica si esta se encuentra activa o inactiva, su título, su descripción, el número de candidaturas, y por último, las fechas de inicio y finalización. Además, si la votación ya ha terminado, se incluyen el porcentaje de los votos obtenido por cada una de las candidaturas.

Para consultar sus detalles, se accede desde la pantalla principal a la lista de votaciones, donde se indican el título y su creador. Para consultar una en específico, el usuario debe pulsar sobre ella en la lista, acción que provoca que se muestren sus detalles, entre los que además se encuentran los resultados si ya ha pasado la fecha de finalización. Este funcionamiento queda recogido en la figura 3.4.

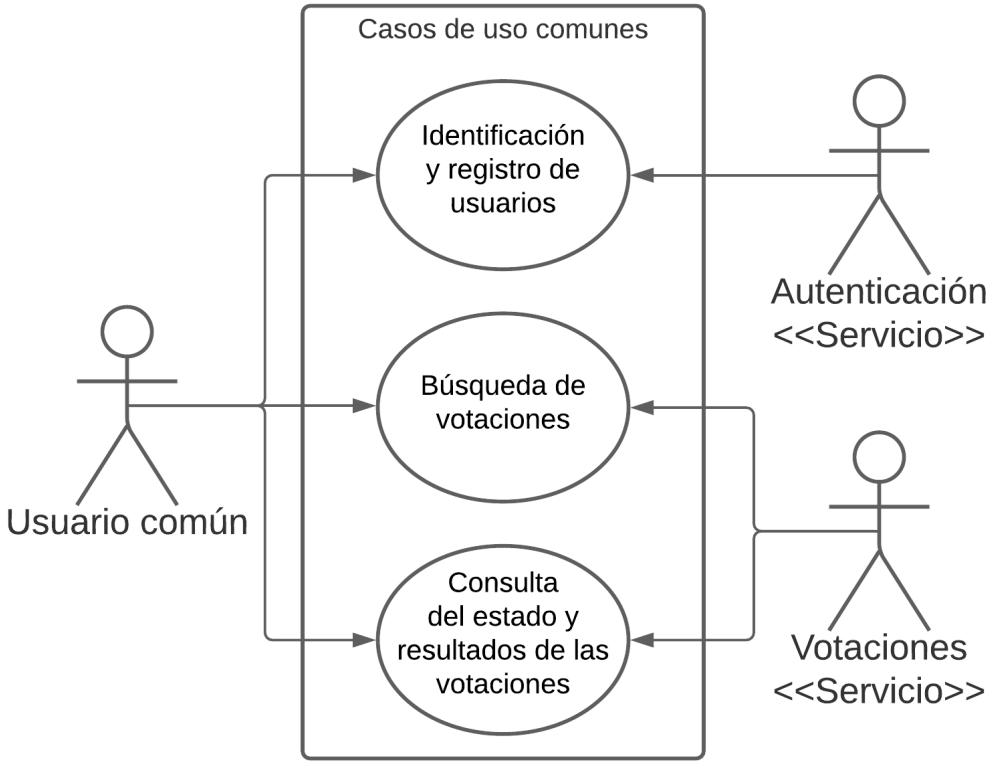

Figura 3.3: Casos de uso del usuario Votante

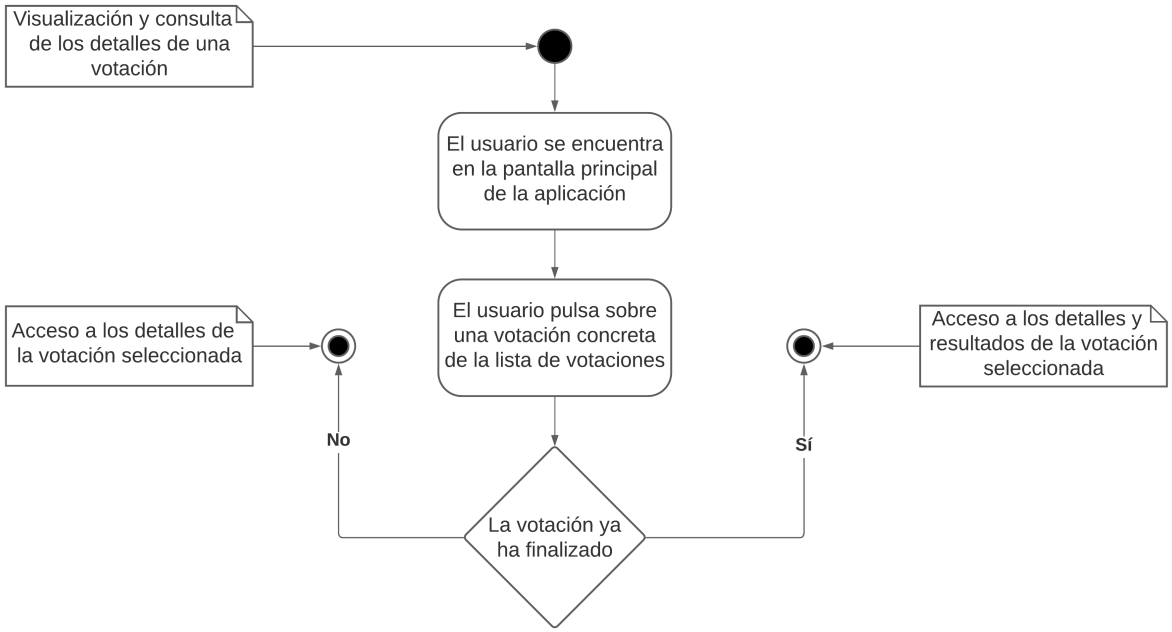

Figura 3.4: Diagrama de flujo de la visualización y consulta de votaciones

Buscar en lista de votaciones

La búsqueda de votaciones se realiza a partir del título de cada una, razón por la que se incluye en la pantalla principal, sobre la lista de votaciones y debajo del apartado de identificación, una barra de búsqueda con la que llevar a cabo esta funcionalidad. Para facilitar la interacción del usuario con el sistema, la búsqueda se realiza cada vez que se introduce o elimina un carácter de la barra, aportando celeridad. Se ilustra en la figura 3.5

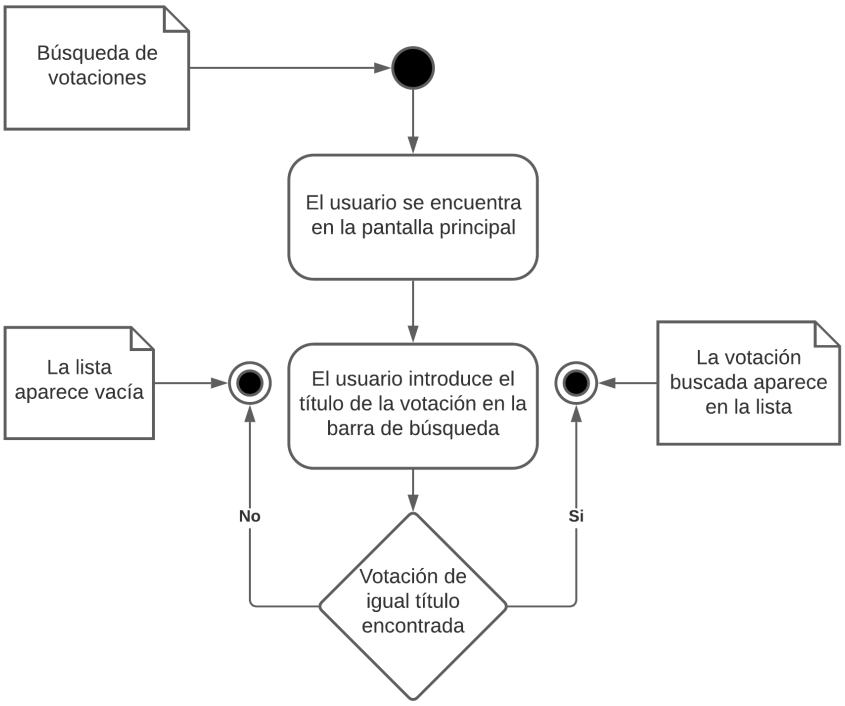

Figura 3.5: Diagrama de flujo de la búsqueda de votaciones

Registro

Para registrar nuevos votantes, la aplicación almacena un nombre de usuario y clave en forma de nombre y hash, que es calculado durante el registro. La diferencia entre votantes y creadores radica en que los primeros pueden registrarse directamente en la aplicación, mientras que los segundos, deben inicialmente ser usuarios registrados y solicitar acceso a las funciones de Creador. Para ello, ingresan en una lista de usuarios pendientes de validación.

El futuro votante debe acceder a la sección registro e introducir en el formulario su elección de nombre y contraseña, para a continuación pulsar el botón “registrarse”. Si ya existe algún otro usuario con el mismo nombre ya registrado, el registro fallará. Si no existe ningún otro usuario con el mismo nombre en el sistema, comenzará el proceso de guardado de los datos del nuevo usuario. Se muestra en la figura 3.6.

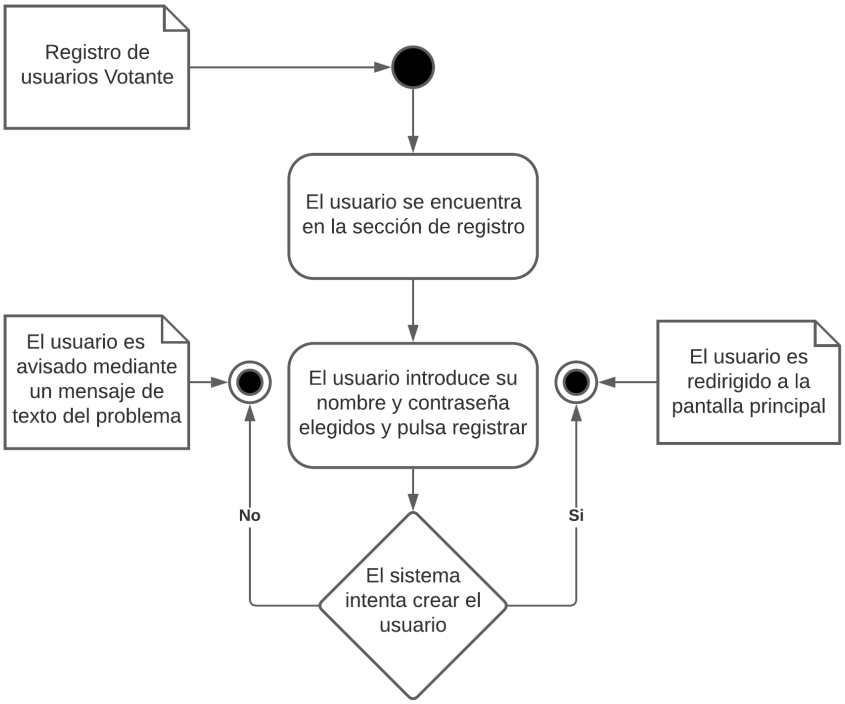

Figura 3.6: Diagrama de flujo del registro de votantes

Identificación de usuario

Este proceso se realiza utilizando dos datos, el nombre de usuario y su contraseña. Todos los usuarios tienen un tipo asociado en función del que se le ofrecen los servicios. La identificación es una sección de la aplicación que dispone de un formulario a rellenar y de un botón de envío. El usuario que desee identificarse, debe introducir su nombre de usuario y contraseña en los campos con mismo nombre del formulario, y a continuación, pulsar el botón de envío para enviar sus datos al servidor. Si el servidor encuentra una coincidencia con estos datos, el usuario accederá, con lo que a partir de ese momento, obtiene acceso a los servicios del tipo que le corresponda. En caso contrario, el usuario es informado del fallo de autenticación mediante un mensaje de texto. Queda ilustrado en la figura 3.7.

3.3.2. Votante

Para realizar las votaciones, previamente se elabora su censo electoral, y únicamente podrán votar sus miembros. Un censo es un grupo de usuarios, y estos deben unirse mediante unas credenciales únicas, que son el nombre del grupo y su clave de acceso. Por ello, los usuarios de tipo votante tienen disponibles tres servicios en el sistema, como se indica en la figura 3.8.

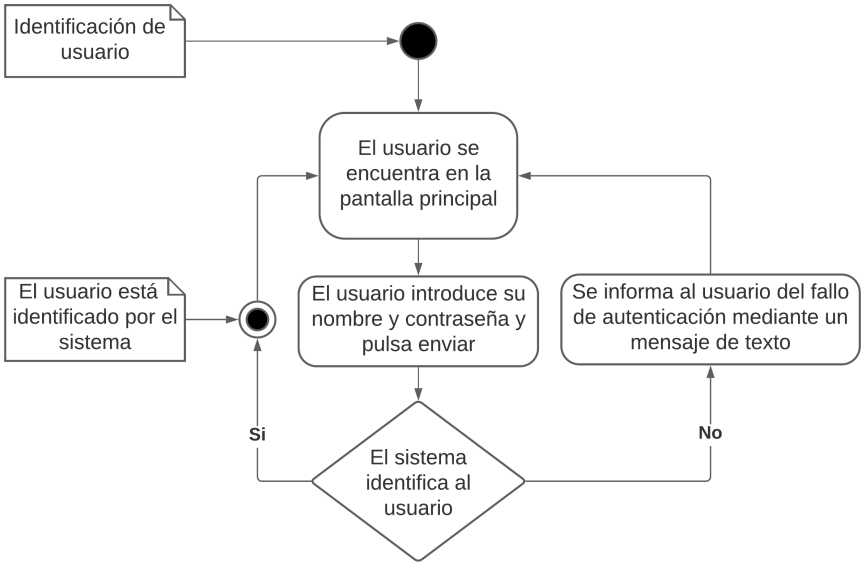

Figura 3.7: Diagrama de flujo de la identificación de usuarios

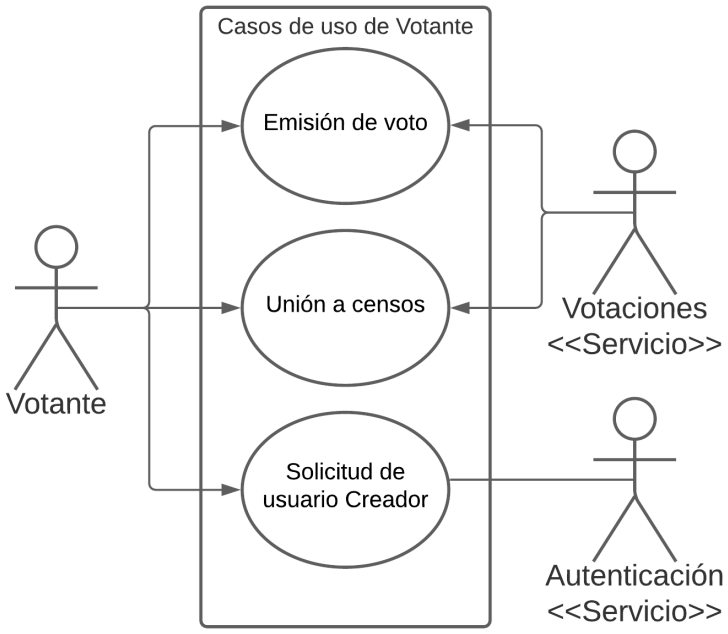

Figura 3.8: Casos de uso del usuario Votante

Ingresar a censo

El usuario, con sus credenciales de unión, accede a la sección de unión a grupos, desde la que dispone de un formulario y un botón de unión. El formulario incluye los campos de nombre y clave de acceso, que una vez rellenados, se envían al servidor mediante el botón de unión. Si son correctos, este pasará a formar parte del censo. En cualquiera de los casos, el usuario será informado con el resultado de su intento de ingreso al censo mediante un mensaje de texto. El proceso es ilustrado en el diagrama 3.9.

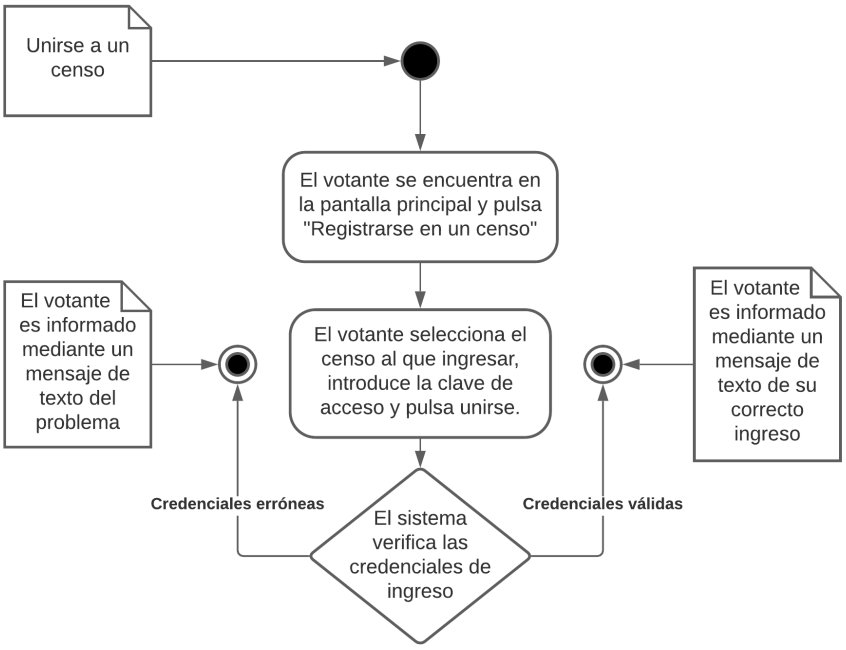

Figura 3.9: Diagrama de secuencia de ingreso a censo

Votar

El usuario debe acceder a la votación en el periodo entre su fecha de inicio y finalización. Si es así, este encuentra un campo donde seleccionar la candidatura, que confirma, y pulsar el botón “Emitir voto”, enviando su solicitud al servidor. Si el voto es inválido, este es descartado. En caso contrario, este se guarda en el sistema sin ningún dato identificativo de su emisor. En ambos casos, el usuario es informado con un mensaje de texto del resultado de su acción. Esto se ilustra en el diagrama 3.10.

Solicitud de usuario Creador

El usuario, desde la pantalla principal, pulsa el botón “Solicitar rango creador”. A continuación, confirma su solicitud, y si esta es enviada, el usuario es informado mediante texto. El proceso es ilustrado en el diagrama 3.11.

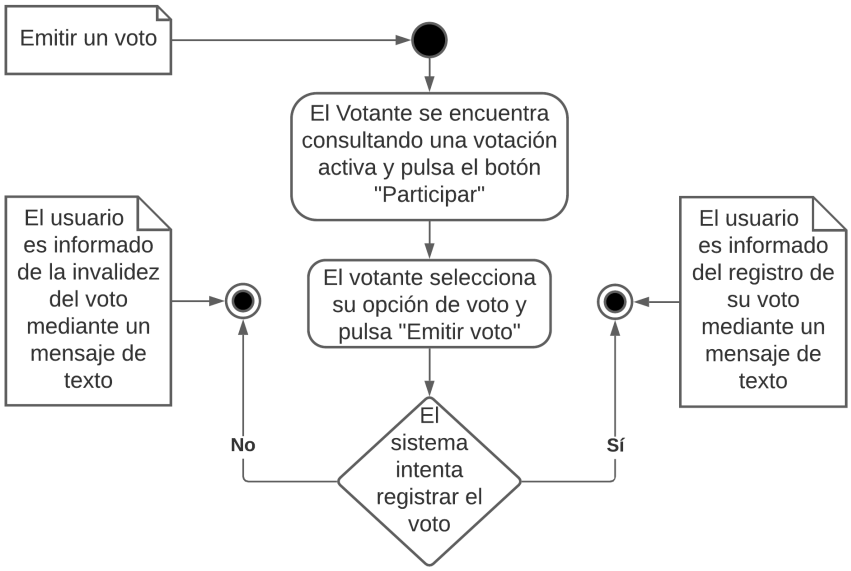

Figura 3.10: Diagrama de secuencia de emisión de votos

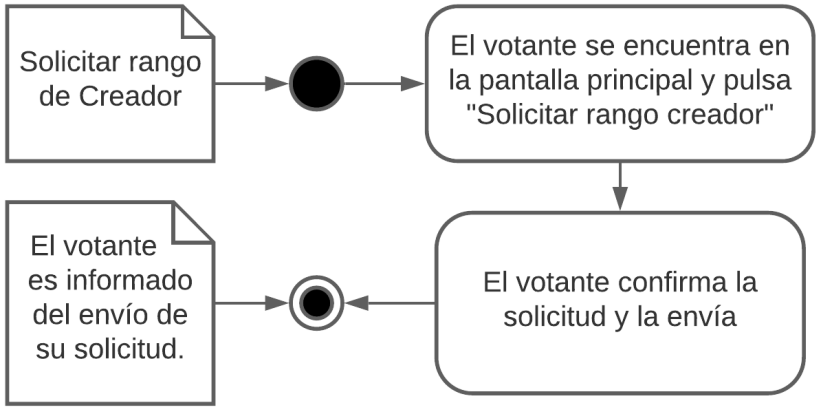

Figura 3.11: Diagrama de secuencia de solicitud de rango Creador

3.3.3. Creador

En el servicio de votaciones electrónicas, los usuarios de tipo creador tienen papel de representantes, ya que son los encargados de crear nuevas votaciones, registrar nuevos grupos de usuarios y añadir opciones de voto. Una vez creado un censo, deben suministrar los datos de acceso a sus miembros para que estos ingresen. Por otro lado, se encargan de establecer todos los parámetros de las votaciones y de su publicación. Sus interacciones se muestran en el diagrama 3.12.

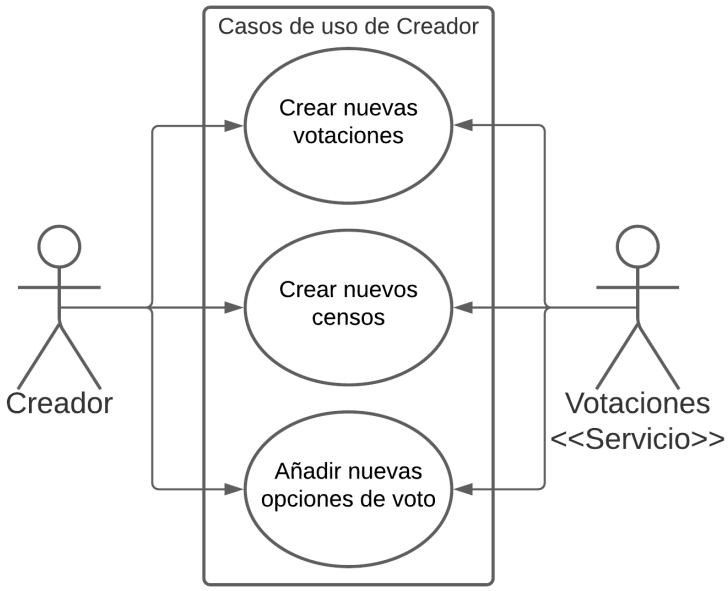

Figura 3.12: Casos de uso del usuario Creador

Nuevo Censo

Un censo es un conjunto de usuarios accesible mediante credenciales. Para crearlos, se necesita especificar un nombre y clave de acceso. Este proceso es ilustrado en la figura 3.13. Esta tarea corre a cargo de un usuario creador, y una vez completa, debe comunicar las credenciales de unión a los miembros para que puedan ingresar. Tras crear un censo, este puede ser seleccionado como censo electoral de las nuevas votaciones.

El usuario debe acceder a la sección de nuevo censo, donde se encuentra un formulario con dos campos, nombre del censo y clave del censo, así como un botón de creación. Una vez son rellenados, se pulsa el botón de creación, lo que envía la solicitud al servidor. Si el servidor lo crea, ese censo pasa a ser seleccionable durante la creación de nuevas votaciones.

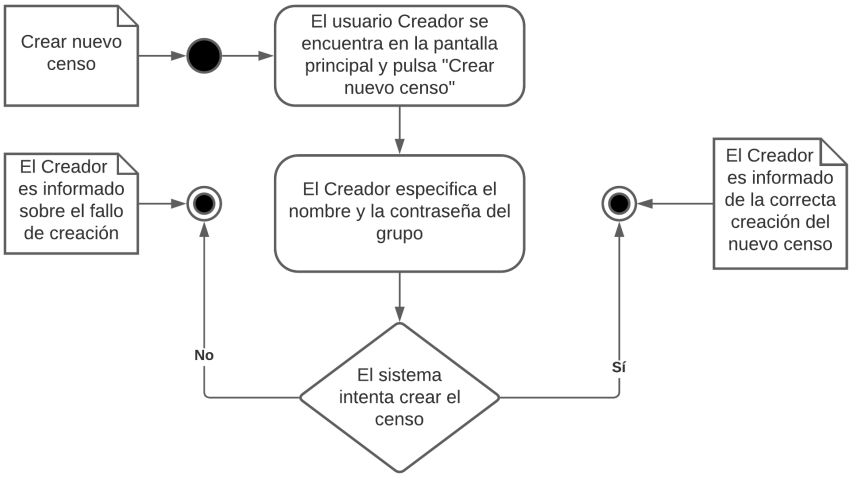

Figura 3.13: Diagrama de secuencia de creación de censo

Nueva votación

Al momento de crear una votación, deben ser especificados título, descripción, fechas de inicio y finalización, y censo electoral. El acceso al voto queda restringido a los miembros del censo electoral. El título y su autor aparecerán en la descripción ofrecida por la lista de la pantalla principal. Desde la fecha de inicio a la de finalización, las votaciones están activas, y en caso contrario, su estado es inactivo. Se considera acabada toda votación pasada su fecha de finalización, realizando en ese momento el escrutinio de los votos, que habilita el anuncio de resultados. El caso de uso queda ilustrado en el diagrama 3.14.

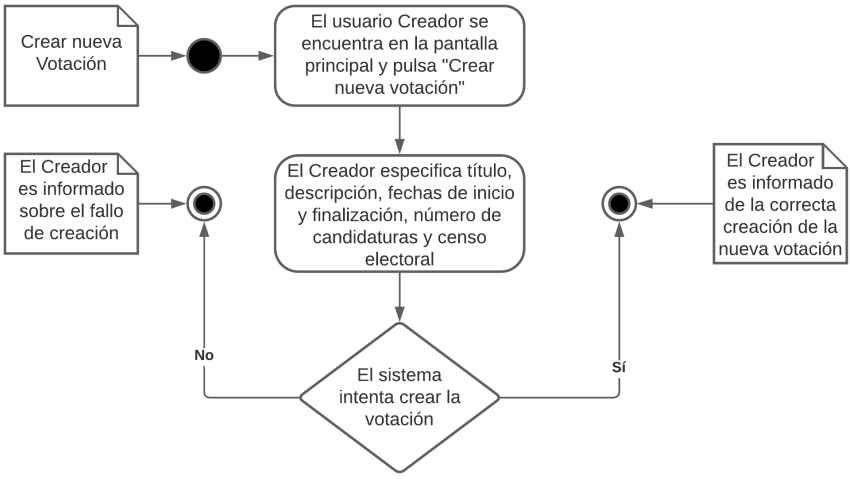

Figura 3.14: Diagrama de secuencia de nueva votación

El creador debe acceder a la sección de nueva votación, donde se encuentra un formulario con

inicialmente cinco campos, título, descripción, fechas de inicio y finalización, y censo electoral. Las candidaturas se añaden desde la consulta de votaciones. Una vez los campos hayan sido completados, el creador deberá pulsar el botón de creación. En caso de ocurrir algún problema durante esta, el usuario será informado mediante un mensaje de texto. En caso contrario, el usuario será redirigido a la página principal, desde la que podrá comprobar que la votación ha sido creada, quedando preparada la infraestructura.

Añadir opción de voto

Si la votación ha sido creada por el mismo usuario que la consulta, este tiene habilitado añadir nuevas opciones de voto. Para ello, deberá pulsar el botón “Añadir nueva opción” en una votación creada por él, y posteriormente especificar el nombre de esta. Esto se ilustra en la figura 3.15

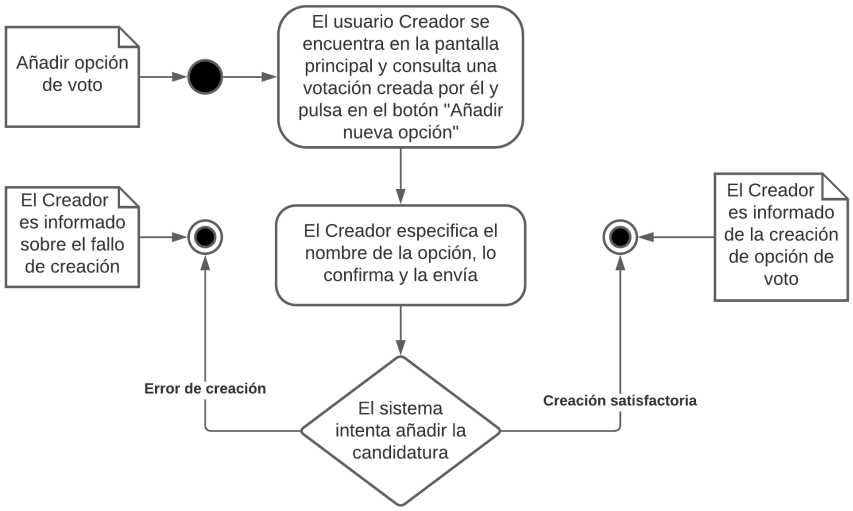

Figura 3.15: Diagrama de secuencia de nueva opción de voto

3.3.4. Administrador

Los administradores tienen la tarea de regular a los usuarios creadores, es decir, a los representantes de los diferentes grupos que utilizan la aplicación. Es con este fin que disponen de una sección donde se muestra una lista con las solicitudes de nuevos usuarios creadores. Es tarea de los administradores validarlos, como se muestra en la figura 3.16.

Una vez se recibe una solicitud, el administrador se debe comunicar con el solicitante y verificar su identidad. Dependiendo del resultado, el administrador dará vía libre a la validación del usuario, mientras que en caso contrario será denegada. Esto se muestra en el diagrama 3.17

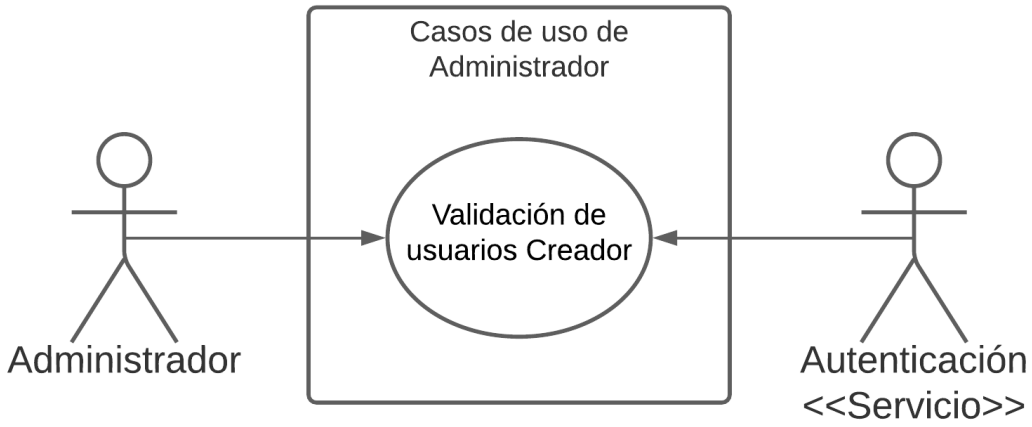

Figura 3.16: Casos de uso del usuario Administrador

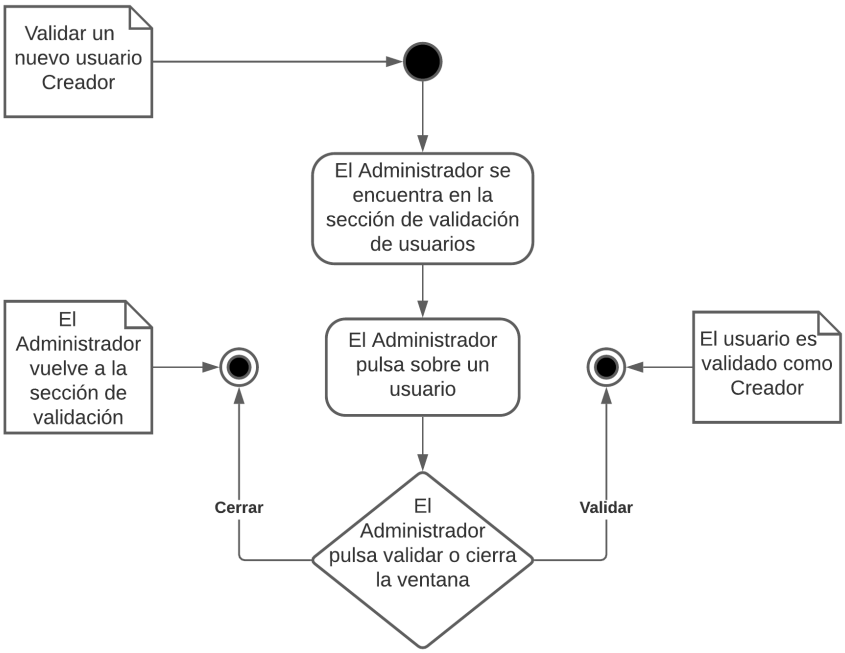

Figura 3.17: Secuencia de validación de usuarios Creador.

3.4. Arquitectura de implementación

A alto nivel, cualquier sistema de verificación puede dividirse en dos componentes, el apartado del cliente y el apartado del servidor. En el primero, los usuarios introducen sus credenciales para generar los votos firmados y enviarlos al servidor. Una vez allí, se verifica su procedencia, y en caso de ser válida, se almacenan para su futuro recuento. Además, debe ser resistente a fallos locales sin alterar el funcionamiento global del sistema, y favorecer su escalabilidad.

Ante estas necesidades, se propone utilizar una arquitectura Modelo-Vista-Controlador (MVC) , que gracias a la separación en módulos, proporcionará un sistema fácilmente escalable. Asimismo, se debe reducir el impacto de los fallos locales, para lo que se propone una base de datos distribuida, que además, dado que el voto debe ser una operación atómica, requiere de transacciones. El modelo de datos queda definido por la figura 3.18.

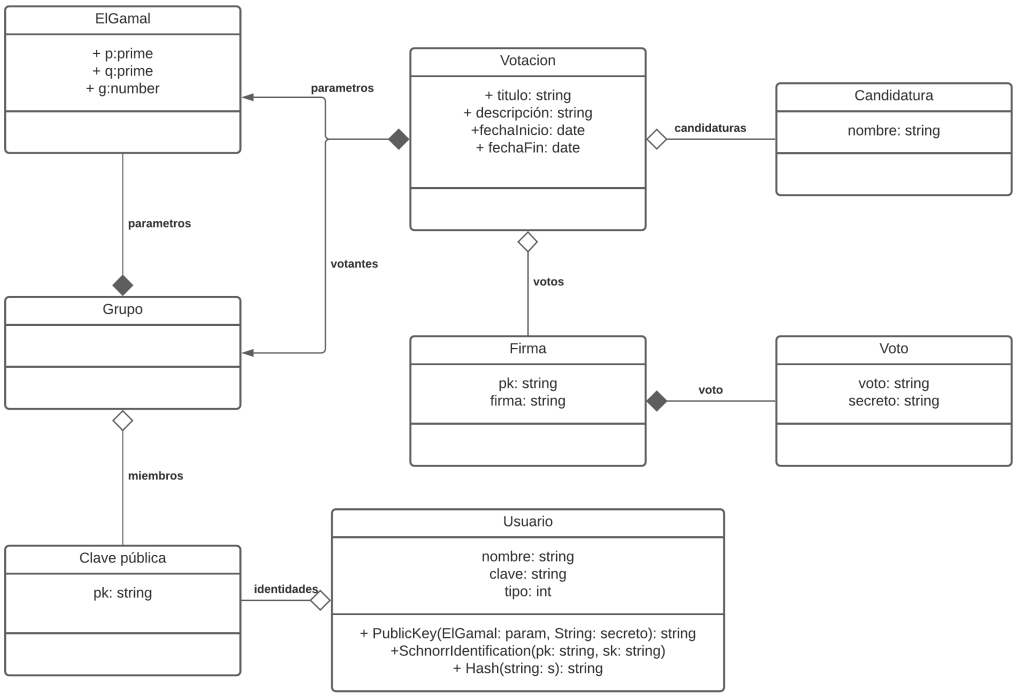

Figura 3.18: Modelo de datos

El servidor y cliente deben acordar un protocolo para almacenar de forma anónima los votos válidos, y como medio de comunicación se implementa una Interfaz de programación de aplicaciones de transferencia de estado representacional (API-REST) , que habilita la interacción entre ambos mediante peticiones del Protocolo de transferencia de hipertexto (HTTP) , posibilitando escalabilidad horizontal del servicio en cualquier momento, lo que además aumenta su tolerancia a fallos. Desde el lado del cliente, esta arquitectura proporciona vistas dinámicas en el cliente a partir de los datos que recibe desde la API, funcionando a tiempo real. Esto se ilustra en la figura 3.19

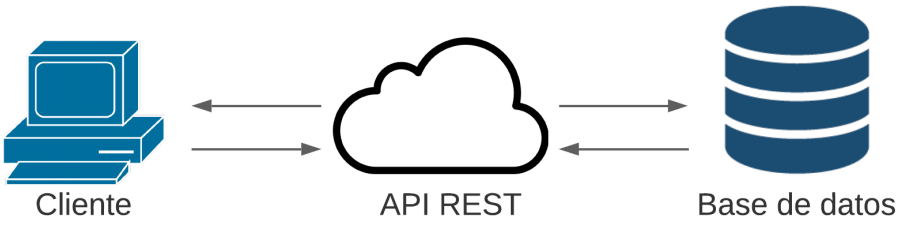

Figura 3.19: Arquitectura del sistema

3.4.1. Tecnologías utilizadas

Un sistema de votaciones debe soportar contextos donde una gran cantidad de usuarios esté votando simultáneamente, por lo que la infraestructura del Modelo debe procesar rápidamente las inserciones de datos, y al mismo tiempo, cada una de las inserciones deben ser atómicas. A esto hay que añadirle que se requiere de un sistema tolerante a fallos, por lo que un modelo distribuido en nodos es una consideración factible. Ante estas necesidades, la base de datos escogida es “MongoDB” [16], y en el capítulo 5 se proponen otras posibilidades.

Para implementar el lado del servidor se utiliza “Django” [10], cuyo código se escribe en “Python” [33], y proporciona una rápida gestión de los modelos, así como las funcionalidades requeridas para implementar una API mediante su librería “Django Rest Framework” [24]. Para la conexión con el modelo de datos, se utiliza la librería “Djongo” [27].

Angular [12], que se escribe en “TypeScript” [26], y proporciona un entorno de trabajo para aplicaciones web tanto del lado del cliente como del servidor, se utiliza en este proyecto para implementar el primero. Para proporcionar una interfaz amigable al usuario, se utiliza la librería Angular Material [13]. Este entorno dispone de una arquitectura modular mediante la separación de las secciones de la aplicación en componentes, y es compatible con la arquitectura API-REST, comunicándose mediante JSON [31].

3.4.2. Votaciones

Para validar votos electrónicos, estos deben enviarse cifrados junto a un token que actúa de clave secreta para validar el proceso de voto. Aunque esté asociado a la clave pública del emisor en cuestión, como esta no contiene información sobre su identidad, el usuario se mantiene anónimo. Contando con la posibilidad del doble voto, se almacenan claves públicas como censo electoral. Cuando un usuario vota, el servicio comprueba que la clave pública no tenga un voto asociado, y si lo hiciera, el sistema anularía el nuevo voto, de lo contrario, el voto queda registrado en el sistema. Esto queda ilustrado en el diagrama 3.20

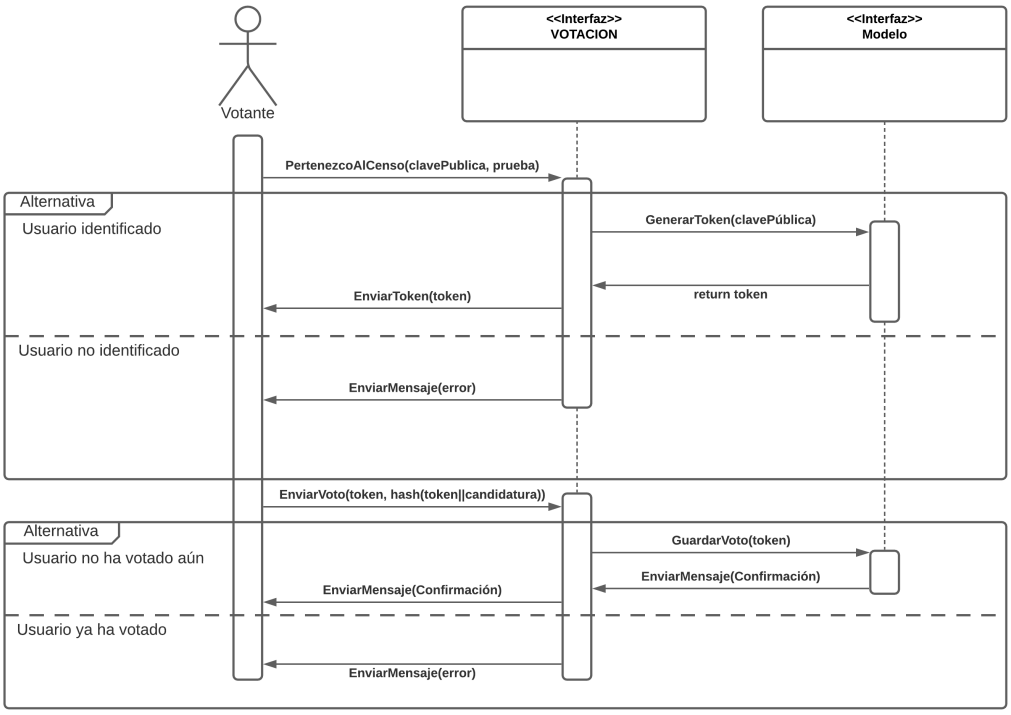

Figura 3.20: Diagrama de secuencia del protocolo de votación

El escrutinio se realiza recorriendo la lista de los votos y tras comprobar la candidatura de cada uno, se suma al recuento de esta, permitiendo a continuación generar las estadísticas del escrutinio y anunciar los resultados, como se muestra en el diagrama 3.21. Como las elecciones de candidatura de cada voto son públicas, aquel usuario interesado en verificar el recuento únicamente necesitará contactar con el servidor para obtener los datos y comenzar el recuento.

Para no vulnerar la anonimidad, al momento de identificarse el usuario, se guarda

$$h = hash(usuario|clave)$$

en variables del cliente, y cuando se une a un grupo, obtiene el nombre n de este y sus parámetros de ElGamal (p_n, q_n, g_n), con lo que genera su secreto mediante $s_n = hash(h|n)$, y con él, calcula su clave pública $[g^{s_n}]_{p_n}$, que se envía al sistema y pasa a figurar como su identidad para ese grupo.

Al momento de votar, el usuario debe demostrar que forma parte del censo, y como el censo es un grupo de usuarios, y estos tienen los parámetros (p_n, q_n, g_n) asociados, la parte del cliente se encarga de calcular su clave pública mediante $[g^{s_n}]_{p_n}$, y si pertenece realmente al censo electoral, iniciar el protocolo de identificación de Schnorr [19], a partir del que demuestra que realmente $[g^{s_n}]_{p_n}$ es su clave pública, sin revelar nada más que la aserción. En ese caso, el sistema le proporcionará un token de acceso que funciona como clave secreta s_t, con la que calcula, una vez ha elegido una candidatura

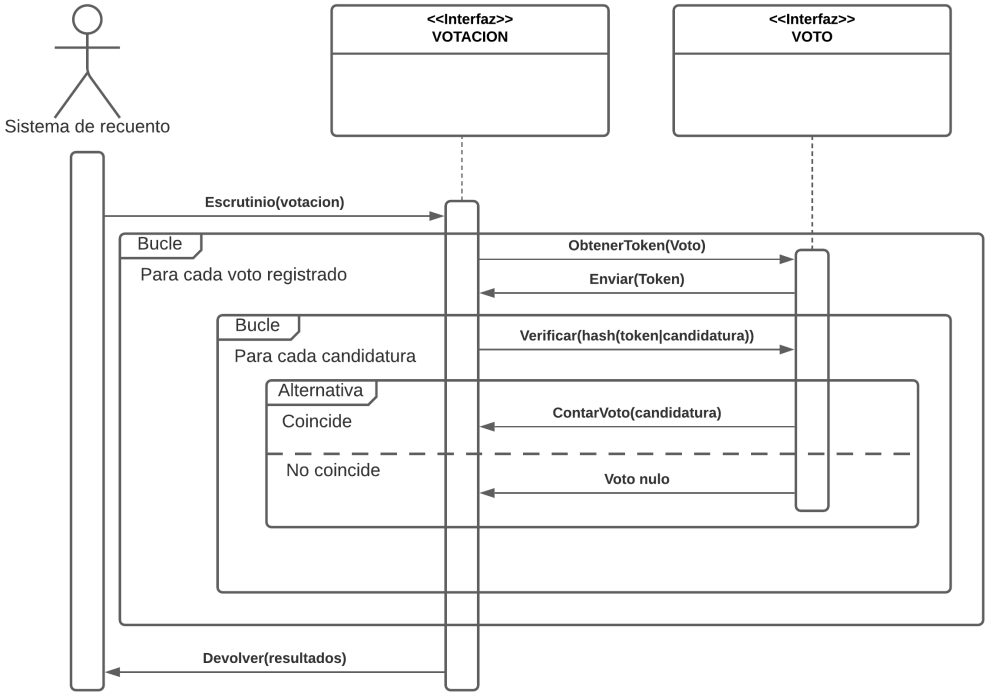

Figura 3.21: Diagrama de secuencia del algoritmo de escrutinio de votos

que llamaremos c, el valor $h_t = hash(s_t||c)$. Una vez dispone de h_t, el usuario enviará al servidor una tupla con los valores (s_t, h_t), que quedará almacenada. En caso contrario, el sistema indicará que se ha producido cierto error.

Durante el escrutinio, el sistema recorre la lista de los votos obtenidos en la votación, y para cada uno ellos, prueba a generar, mediante el token correspondiente al voto s_t, para cada posible candidatura c_i, el valor $h = hash(s_t||c_i)$. En caso de coincidir con el asociado al token h_t, significa que el usuario ha votado por esa candidatura, por lo que el número de votos de esta aumenta en uno. El proceso continúa hasta agotar la lista, generando estadísticas y anunciando los resultados.

Doble voto

Tanto en implementaciones de dinero como votaciones, existe el peligro del doble gasto [1]. Esto suceso tiene lugar cuando una misma moneda digital es gastada dos veces, y en el caso de las votaciones, se sustituye la moneda por las credenciales que otorgan el derecho a sufragio. Para evitarlo, se debe disponer de un mecanismo de control de gastos, que en el ámbito electoral corresponde a la lista de claves públicas que identifican anónimamente a los votantes.

Para votar, el usuario debe obtener un token de autenticación, cuya privacidad se analiza en la sección 3.4.2. Una vez dispone de este token, el usuario lo hashea junto a la opción escogida y lo envía al servidor. Si este intentase votar de nuevo, al identificarse como un miembro del censo, pero

encontrarse un token de autenticación asociado a este, el usuario no podría hacerlo, ya que el sistema lo denegaría.

El usuario recibe un token de autenticación generado aleatoriamente por el servidor y asociado a su clave pública censal, lo llamaremos $s = random()$. Mediante s, el usuario introduce la candidatura escogida c en una tupla (s, c), y la envía al servidor. En este, si s está en la lista de tokens generados y no hay ningún voto registrado con ella, se guarda el voto $v = hash(s||c)$.

Como la validación del voto depende de su pertenencia a la lista de usuarios del censo, y la garantía de que este aún no ha votado, supongamos que esté en el censo y ya haya votado. En ese caso, se ha registrado en la aplicación el envío de su voto, por lo que cuando se reciba otro de la misma clave pública para la misma votación, como esta se repite, el sistema detecta que ya ha votado, razón por la que se descarta, imposibilitando el doble voto, como se mostraba en la figura 3.20.

Voto anónimo y escrutinio público

Al momento de ingresar en un grupo, el usuario, cuyo nombre y contraseña se representan mediante las variables $user$ y $pass$, calcula el resultado de la función hash $s_k = hash(user||pass)$, utilizándolo a la hora de ingresar en el grupo con parámetros de ElGamal (p_g, q_g, g_g) para generar el identificador $id = p_k = [g_g^{s_k}]_{p_g}$, siendo este la clave pública a partir de la que el usuario se identifica en el grupo.

Para votar, el usuario debe identificarse como miembro del censo, y para ello utiliza el protocolo de identificación de Schnorr, de forma que como él conoce la clave secreta s_k de su identidad pública p_k en el grupo censal, y mediante este protocolo, estudiado en la sección A.2.3, es capaz de demostrar su conocimiento sin revelarla, la privacidad del usuario se mantiene intacta durante el proceso de identificación. Una vez se ha identificado, el servidor genera aleatoriamente un token t y obtiene las variables de ElGamal (p_v, q_v, g_v) de la votación. A partir de estas, crea una firma $f = [g_v^{s_k}]_{p_v}$, y se la asocia a la clave pública p_k del usuario, para a continuación responderle con un mensaje indicando identificación correcta, y acompañado del token t. Al recibirlo, este elige una candidatura c, produce el voto mediante $v = hash(t, c)$, y a continuación responde al servidor con una tupla (t, v), que el servidor recibe. Al comprobar que se trata del token del usuario y que este no ha votado aún, registra en el sistema.

Al momento de realizar el recuento de votos v_i, estos se publican junto a los tokens t_i, y a su vez, se publica la lista de claves públicas p_{ki} y las firmas f_i. La verificación es capaz de asociar v_i a t_i, t_i a f_i, y esta última está asociada a la clave pública $p_k i$ del usuario. Esto no supone un problema porque el poseedor del secreto de la clave pública $p_k i$ demuestra su conocimiento mediante el protocolo de identificación de Schnorr, razón por la que no se dispone de información asociada a su identidad real, volviendo el proceso de voto anónimo y de recuento público.

4

Pruebas

Durante este capítulo se verifica la funcionalidad de la aplicación. Para ello, se utilizan pruebas de ingeniería del software. Las pruebas realizadas consisten principalmente en pruebas unitarias, pruebas de integración y pruebas del sistema, finalizando con las pruebas de aceptación del cliente.

Las pruebas unitarias verifican la funcionalidad del sistema por unidades. Se realizan pruebas de caja negra en cada uno de los módulos que componen la aplicación. Cuando todos estos las pasan satisfactoriamente, dado que gran parte del servicio requiere de la interacción entre módulos, se desarrollan pruebas de caja blanca de los casos de uso de la aplicación. Esto garantiza el correcto flujo de información del que requieren las acciones de los usuarios. La interfaz de la aplicación, desarrollada en Angular, se ha probado con orientaciones horizontales y verticales, manteniéndose clara a la vista en ambos casos. El servidor y la base de datos han sido probados mediante el propio módulo de administración de la herramienta Django.

Las pruebas de integración verifican la cohesión entre los módulos de la aplicación. Se ha comprobado la correcta comunicación entre Angular [12] y Django [10] mediante el análisis de las peticiones y respuestas. Mediante diferentes formularios y la herramienta Postman [29], se verifica el correcto funcionamiento de la API-REST, enviando peticiones a los diferentes endpoint y comparando el efecto de la interacción.

Las pruebas del sistema verifican la integridad de su información. Se han intentado ejecutar ataques SQL Injection y Cross-site scripting (XSS) , pero ninguno de ellos ha resultado efectivo. La aplicación ha sido probada satisfactoriamente en los exploradores web Brave Browser, Google Chrome, Safari y Microsoft Edge.

4.1. Pruebas de aceptación

Se le muestra al cliente una ejecución completa de la funcionalidad de la aplicación, y este da su aprobación sobre el proyecto desarrollado. Esta prueba, comienza con un usuario no identificado en la página principal, como se muestra en la figura 4.1.

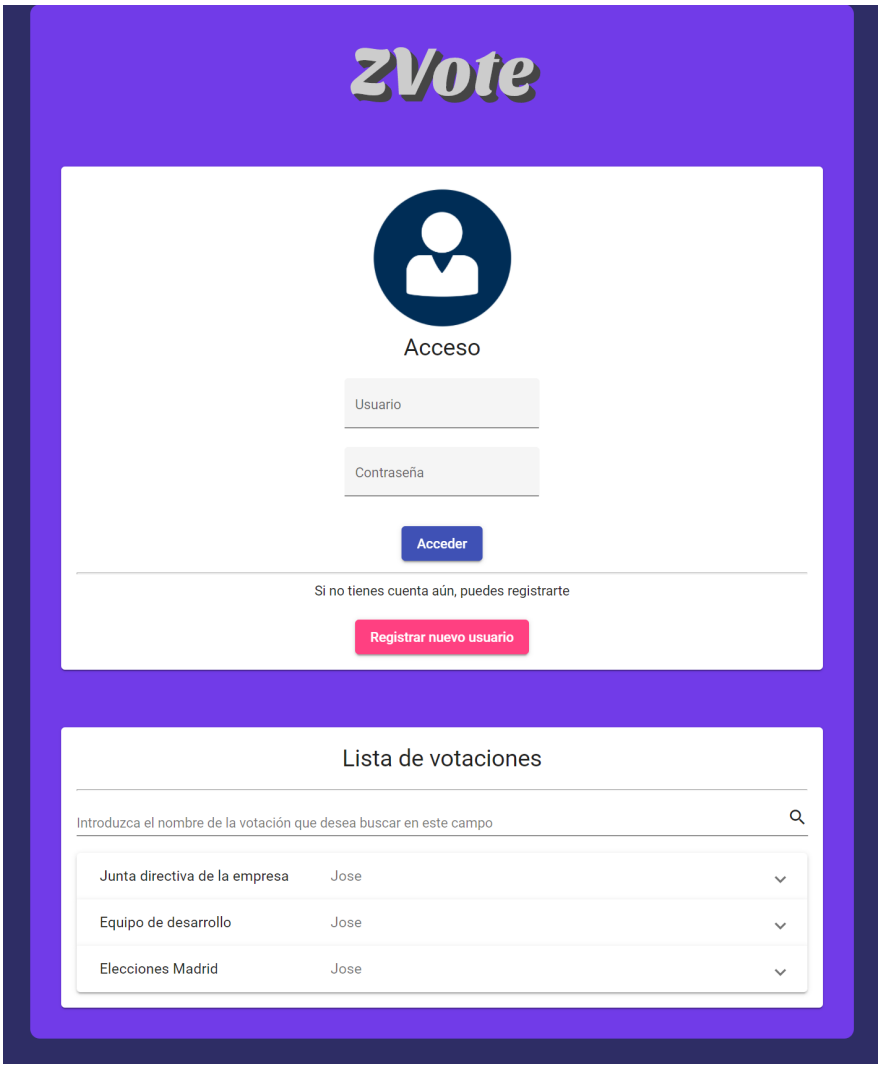

Figura 4.1: Página de inicio de la aplicación.

Como este no tiene cuenta, debe registrarse rellenando para ello el formulario de las figura 4.2, al que accede pulsando en el botón de “Registrar nuevo usuario”. A continuación, rellena el formulario de acceso y pulsa “Acceder”.

Figura 4.2: Formulario de registro en la aplicación.

Su siguiente tarea es solicitar el rango Creador, tras la que ingresa en un censo, y vota en una votación en la que participa el censo al que acaba de acceder. De esta forma, se prueba toda la funcionalidad desde la vista de usuario Votante, reflejada en la figura 4.4.

Una vez le llega la solicitud de Creador, un administrador contacta con el usuario y le valida como Creador. Ya validado, el usuario puede crear votaciones y censos. Inicialmente crea un censo y una votación entre sus miembros mediante los botones “Crear nuevo grupo” y “Crear nueva votación” respectivamente. Inmediatamente, añade a la votación varias opciones mediante el botón “Añadir nueva opción”. A continuación, envía las credenciales de acceso al censo a aquellos usuarios que deben votar. El usuario Creador realiza estas acciones a través de la pantalla de la figura 4.5

Varios usuarios ingresan en el censo y votan en la votación del usuario, rellenando el formulario de la figura 4.6, que si se realiza correctamente lleva a la pantalla mostrada en la figura 4.7.

La votación finaliza, y se comprueba que cualquier cliente que acceda a la aplicación tiene dispo-

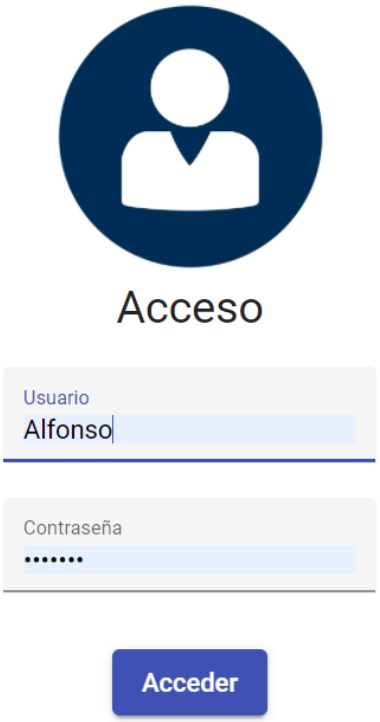

Figura 4.3: Formulario de acceso a la aplicación.

Figura 4.4: Página principal de usuarios Votante.

Figura 4.5: Página principal de usuarios Creador.

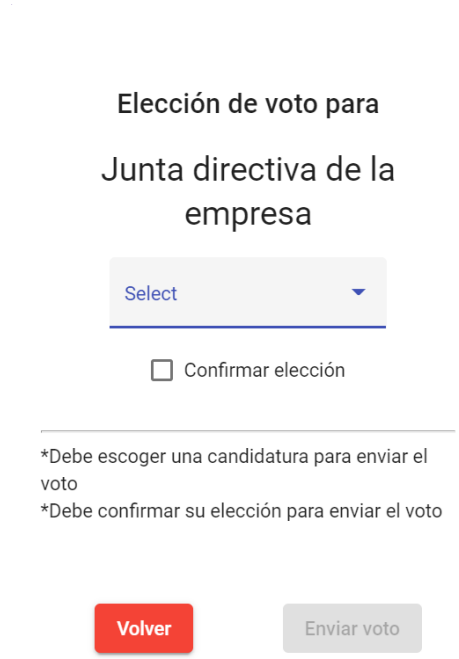

Figura 4.6: Pantalla de emisión de voto.

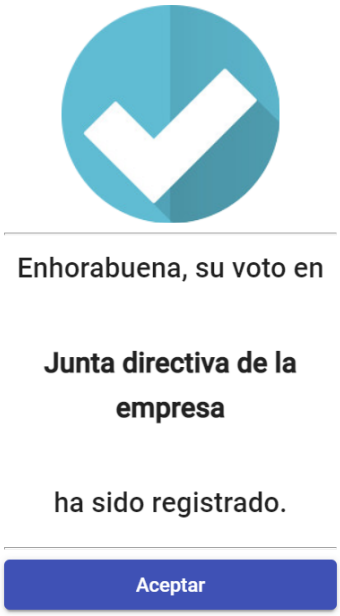

Figura 4.7: Pantalla de emisión de voto correcto.

nible la visualización del resultado, como se ve en la figura 4.8.

Figura 4.8: Resultados de la votación.

5

Conclusiones

La finalidad de este proyecto es proporcionar un servicio de votaciones electrónicas de uso general, para lo que se debe garantizar la imposibilidad del doble voto y la anonimidad del votante. Se han utilizado para ello pruebas de conocimiento cero, que en el primer caso nos permiten detectar la duplicidad de voto de un emisor por la aparición de su clave pública entre los ya recogidos. En el segundo, estas nos proporcionan una infraestructura donde la clave pública no aporta información alguna sobre la privada, manteniendo al emisor del voto imposible de identificar.

Este sistema puede simular las fases de registro, voto y recuento de cada votación. La primera se debe a los usuarios de tipo Creador, que se encargan de registrar nuevas votaciones, especificando su título, descripción, fechas de inicio y finalización, candidaturas y censo electoral. La segunda gira en torno a sus votantes, garantizándoles almacenamiento anónimo del voto mediante pruebas de conocimiento cero, así como un mecanismo de autorización para evitar el doble gasto basado en claves públicas y privadas. Por último, se realiza el escrutinio públicamente, sin vulnerar en ningún momento la privacidad de los votantes, gracias a que las claves públicas almacenadas para el grupo en cuestión no proporcionan dato alguno sobre la identidad real del usuario.

Durante su desarrollo, aparece la necesidad de modularidad, aportando independencia entre componentes. Esto aporta escalabilidad y soporte a nuevas funcionalidades. La modularidad en el sistema de almacenamiento hace pensar en una distribución por nodos, que si mantienen cierto grado de redundancia y se comunican entre ellos, proporcionan un entorno estable donde almacenar las votaciones finalizadas, pues aunque las que aún no lo han hecho requieren de inserciones de nuevos datos, las que sí, únicamente deben conservar la integridad de estos. Ante la posibilidad de aumentar el alcance de la aplicación, el sistema de identificación puede ser mejorado, soportando más tipos de documentos identificativos, para así generalizar la aplicación a todo tipo de ámbitos.

De esta forma, se comprueba que a la hora de diseñar sistemas de votaciones hay que abordar una gran problemática, sin embargo, partiendo de un canal de comunicación seguro y de un entorno con bajo riesgo de coacción, hemos comprobado que es posible llegar a un sistema que aborde las cuestiones propuestas en los objetivos del proyecto. Se presentan dos potenciales mejoras sobre los componentes de la arquitectura.

5.1. Trabajo futuro

Los sistemas de voto electrónico añaden un reto de seguridad ante la posibilidad de un tercero obteniendo las credenciales de cierto votante y votando en su lugar. Es objeto de estudio la necesidad de un recuento de votos transparente para proporcionar verificación de resultados a sus usuarios. Esto sugiere la adopción de un sistema de almacenamiento que añada dependencias a las nuevas entradas respecto de las anteriores, y que al mismo tiempo, gracias a su naturaleza distribuida, implemente un mecanismo de consenso para validar nuevos votos en la red, manteniendo anónimos a sus emisores. Asimismo, un sistema de voto electrónico que utilice documentos de identificación electrónica, no requeriría de nombres de usuario y contraseñas, necesitando sólo un lector de tarjetas electrónicas.

Identificación electrónica

A la hora de resolver el problema de la identificación correcta de usuarios, la solución más prometedora es la utilización de una clave pública única para cada usuario como puede ser la del Documento Nacional de Identidad electrónico (DNIe) . Si este sistema fuese aprovechado [17], los ciudadanos dispondrían de la posibilidad de utilizar su documento de identidad con el fin de demostrar que son una persona con derecho a sufragio en unas elecciones arbitrarias. Sin embargo, esta solución se encuentra fuera del alcance del proyecto.

Otra opción es utilizar ciertos factores biométricos [18] de los usuarios, como su iris para generar una clave privada, mediante cuya clave pública, pueda interactuar con el sistema. Así, el peso del proyecto caería sobre el diseño de un escáner capaz de detectar las suficientes diferencias individuales como para convertirlo en un mecanismo de identificación válido para su uso en todo tipo de votaciones.

Blockchain

La privacidad e integridad de los datos es intrínseca al proceso electoral, razón por la que se proponen arquitecturas de cadena de bloques, generando dependencia de los votos anteriores y añadiendo nuevos registros en función de una función hash criptográfica. Los datos más afectados por esta arquitectura son las elecciones de candidatura y sus secretos, con los que se demuestra la validez del voto en cuestión. La dependencia de los datos, junto a una distribución de la red donde todos los nodos reciben los avisos de cada nuevo voto, define un consenso a la hora de añadir nuevas entradas a la cadena de bloques, es decir, a la hora de validar nuevos votos. Se debe escoger un mecanismo de selección de nodos rápido y seguro [5], sin embargo, son aspectos a balancear, ya que la mejora de uno viene generalmente a expensas de la garantía sobre el otro. Se busca un mecanismo de validación de nuevos votos verificable, y un criterio de elección aprobado mediante consenso que garantice la integridad de la información almacenada, sin revelar nada sobre sus emisores [4].

Bibliografía

[1] B. Academy, *¿qué es el doble gasto?*, 2021. (Descargar).

[2] B. Adida, *Helios: Web-based open-audit voting.*, 01 2008, pp. 335–348.

[3] M. Bellare and P. Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, CCS '93, New York, NY, USA, 1993, Association for Computing Machinery, p. 62–73.

[4] E. Ben-sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, *Zerocash: Decentralized anonymous payments from bitcoin*, 05 2014, pp. 459–474.

[5] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, *Succinct non-interactive zero knowledge for a von neumann architecture*, in Proceedings of the 23rd USENIX Conference on Security Symposium, SEC'14, USA, 2014, USENIX Association, p. 781–796.

[6] J. Benaloh, R. Rivest, P. Ryan, P. Stark, V. Teague, and P. Vora, *End-to-end verifiability*, (2015).

[7] D. Chaum, *Blind signatures for untraceable payments*, in Advances in Cryptology, D. Chaum, R. L. Rivest, and A. T. Sherman, eds., Boston, MA, 1983, Springer US, pp. 199–203.

[8] J. del Estado, *Ley orgánica 5/1985*, 1985. (Descargar).

[9] P. D. for Citizens' Rights and C. Affairs, *Potential and challenges of e-voting in the european union*, 2016. (Descargar).

[10] D. S. Foundation, *Django*, 2021. (Descargar).

[11] C. Generales, *Constitución española*, 1978. (Descargar).

[12] Google, *Angular*, 2021. (Descargar).

[13] ——, *Angular material*, 2021. (Descargar).

[14] M. Green, *Zero knowledge proofs: An illustrated primer*, 2014. (Descargar).

[15] E. B. Inc., *Election audit*, 2014. .

[16] M. Inc, *Mongodb*, 2021. (Descargar).

[17] C. D. R. A. I. D. S. (INTECO-CERT), *GuÍa para desarrolladores con el dni electrÓnico*, 2020. (Descargar).

[18] A. Jain, A. Ross, and S. Prabhakar, *An introduction to biometric recognition*, IEEE Transactions on Circuits and Systems for Video Technology, 14 (2004), pp. 4–20.

[19] M. Just, *Schnorr Identification Protocol*, Springer US, Boston, MA, 2011, pp. 1083–1083.

[20] Kaspersky, *Polys – the digital voting system*, 2020. .

[21] W. Kluwer, *Derecho de sufragio activo*, 2020. (Descargar).

[22] D. Kundro, *Criptografía homomórfica: un paradigma de cifrado cada vez más cercano*, 2019. .

[23] N. Li, 2005. (Descargar).

[24] E. O. Ltd, *Django rest framework*, 2021. (Descargar).

[25] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 2001.

[26] Microsoft, *Typescript*, 2021. (Descargar).

[27] nesdis, *Djongo*, 2021. (Descargar).

[28] Polkadot, *Polkadot*, 2021. .

[29] Postman, *Postman*, 2021. .

[30] K. Salamonsen, *A security analysis of the helios voting protocol and application to the norwegian county election*, 2014. .

[31] S. Software, *Javascript object notation*, 2017. (Descargar).

[32] J. R. Vacca and J. R. Vacca, *Computer and Information Security Handbook, Second Edition*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd ed., 2013.

[33] G. van Rossum, *Python*, 2021. (Descargar).

Definiciones

endpoint Punto de acceso a una API.

e-voting Votaciones electrónicas.

SQL Injection Método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar operaciones sobre una base de datos..

Acrónimos

API Interfaz de programación de aplicaciones.

API-REST Interfaz de programación de aplicaciones de transferencia de estado representacional.

DNIe Documento Nacional de Identidad electrónico.

HTTP Protocolo de transferencia de hipertexto.

MVC Modelo-Vista-Controlador.

UML Lenguaje unificado de modelado.

XSS Cross-site scripting.

Apéndices

A

Criptografía para votaciones electrónicas

El principal problema de las votaciones remotas es la presencia de terceras personas observando al usuario mientras emite el voto, descubriendo su elección, así como la coerción durante el voto. Los esquemas de este tipo, funcionan bastante bien en entornos donde los riesgos mencionados son mínimos, maximizando los beneficios del uso de tecnologías durante el proceso electoral.

En los esquemas de clave pública, se pueden cifrar mensajes a partir de esta, y serán únicamente descifrables por el poseedor de su clave secreta. Si el esquema proporciona aditividad homomórfica, el voto cifrado puede añadirse a una única cadena de caracteres, que constituye los resultados de la votación, y que es descifrada únicamente durante la fase de escrutinio, garantizando que ningún voto individual es revelado. Con el fin de proporcionar servicios de votaciones electrónicas verificables, se exploran las siguientes técnicas criptográficas: Distribución de secretos, umbral y pruebas de conocimiento cero.

A.1. Distribución de secretos

En las técnicas de compartición de secreto y umbral, el secreto es distribuido por partes entre varios agentes, que trabajan conjuntamente para recuperar su totalidad. Mientras que en las primeras es necesario que una autoridad emita la información secreta a distribuir, en las segundas no es necesaria, y los agentes pueden trabajar conjuntamente para generar la información secreta y distribuirla. Estas técnicas están basadas en el esquema de compartición de secretos de Shamir, que consiste en dividir un conjunto de datos en n partes, asegurando que el conocimiento de k o más partes permita recuperar la información en su totalidad, mientras que menos de k partes dejen su valor indeterminado [32].

Su funcionamiento está basado en la necesidad de $n+1$ puntos para definir un polinomio de grado n. Si se conoce un secreto S y se desea distribuir entre n miembros, se generan números aleatorios a_i a partir de $i = 1, 2, ..., n$, obteniendo el polinomio

$$f(x) = S + a_1 * x + a_2 * x^2 + ... + a_n * x^n$$

A continuación, se construyen las diferentes partes $(i, f(i))$, y se distribuyen entre los miembros,

aportando cada una de ellas una solución parcial que permite, mediante interpolación polinómica de Lagrange, obtener el secreto S.

En el criptosistema ElGamal con umbral, n participantes se ponen de acuerdo en los parámetros de ElGamal (p, q, g), donde p y q son números primos, se cumple que $q|p-1$, y g es un elemento de orden q del conjunto G_p^*, es decir, $[g^q]_p = 1$. Al comienzo, estos eligen aleatoriamente n valores x_i y cada uno de ellos calcula, a partir de su elección, $y_i = g^{x_i}$. La clave pública y se genera mediante la multiplicación de sus partes, es decir, $y = \prod_i y_i$, por lo que ahora es conocida públicamente. Sin embargo, no pueden obtener la clave secreta x a menos que todos colaboren, para lo que se debe distribuir x de forma verificable a todos los participantes tal que cualquier grupo de k participantes pueda recuperarla.

A.2. Pruebas de conocimiento cero

Las pruebas de conocimiento cero permiten demostrar el conocimiento de un dato sin necesidad de revelarlo, para lo que debe cumplir tres propiedades [14]. La primera es que si el agente es honesto y el verificador también, el agente podrá convencer al verificador con probabilidad abrumadora. La segunda consiste en que si un agente deshonesto intenta convencer a un verificador honesto, la probabilidad de hacerlo es ínfima. La última, conocimiento cero, implica que un verificador deshonesto no obtendrá más información del agente honesto que la veracidad de su demostración.

Según “Handbook of Applied Cryptography” [25], el sistema debe garantizar tres propiedades, la primera es que si tanto el agente como el verificador son honestos, el protocolo será satisfecho con un error mínimo, que se establece dependiendo de su utilidad práctica. La segunda, que si existe un algoritmo de tiempo polinómico probabilístico tal que un agente deshonesto puede, con probabilidad significativa, ejecutar correctamente el protocolo con un verificador honesto, el mismo algoritmo debe ser capaz de extraer conocimiento equivalente al secreto del agente honesto. Por último, la propiedad de conocimiento cero pide que exista un algoritmo probabilístico en tiempo polinómico que si recibe como entrada lo que debe validar, sus interacciones serán indistinguibles de la interacción con el agente honesto original, por lo que el verificador no aprende nada sobre el emisor durante el proceso.

A.2.1. Alí Babá

Alice y Bob se encuentran en la entrada de una cueva, y hay un camino, que al adentrarse en ella, se bifurca en dos ramas, izquierda y derecha. Bob quiere demostrar a Alice que hay una puerta con un código que ella conoce que te permite, si entras por la rama izquierda, salir por la derecha, rodeando por el interior de una cueva, pero Bob no quiere que Alice la vea, por lo que debe demostrárselo de otra forma.

Para ello, Bob propone a Alice cerrar los ojos y esperar a que ella haya entrado en la cueva. En ese momento, Alice deberá indicarle por qué camino salir. Por cada repetición de este proceso, la probabilidad de que Bob esté mintiendo se reduce en un factor $\frac{1}{2}$, ya que hay 2 posibilidades en cada intento. Se recoge una descripción a alto nivel a continuación:

```
Data: Alice, Bob, Cueva, Código secreto
Result: Veracidad de la afirmación de Bob
probabilidadMentir = 1;
while probabilidadMentir ≥ 0,0...1 do
    CerrarOjos(Alice);
    direccionEntrada = ElegirDireccion(Bob);
    Entrar(Bob, Cueva, direccionEntrada);
    AvisarEstoyPreparado(Bob);
    AbrirOjos(Alice);
    direccionAlice = ElegirDireccion(Alice);
    Avisar(direccionAlice, Bob);
    if direccionEntrada != direccionAlice then
        Introducir(Bob, Código secreto);
        direccionSalida = direccionAlice;
    else
        direccionSalida = direccionEntrada;
    end
    Salir(Bob, Cueva, direccionSalida);
    if direccionAlice != direccionSalida then
        return false;
    else
        probabilidadMentir = probabilidadMentir / 2;
        if probabilidadMentir ≥ 0,0...1 then
            return true;
        else
        end
    end
end
```

Algoritmo A.1: Prueba de conocimiento cero básica

A.2.2. Heurísticas de Fiat-Shamir

Como las demostraciones interactivas de conocimiento cero tienen una estructura generalmente común, se pueden transformar en demostraciones no interactivas, mediante lo que se conoce como

heurísticas de Fiat-Shamir. Una prueba de conocimiento cero suele describirse, a alto nivel, como una interacción entre agentes tal que:

1.– El agente A envía un primer mensaje x al verificador V con un dato aleatorio.

2.– V le responde con otro dato aleatorio c.

3.– A responde r, en cuyo cálculo intervienen x, c y el secreto s del agente.

Las demostraciones interactivas requieren de la participación activa del agente y verificador durante la ejecución del protocolo. Como existen situaciones en las que esta participación sea problemática, es conveniente disponer de una herramienta que permita transformarla en una demostración no-interactiva. La prueba de conocimiento cero no interactiva asociada a la prueba anterior es la que sigue:

1.– El agente A genera x.

2.– A introduce x y alguna otra información necesaria para la demostración en inputs de una función hash y llama a la salida c.

3.– Por último, A calcula r y envía (x, c, r) al verificador V.

La seguridad de las pruebas de conocimiento cero no interactivas es demostrable mediante el modelo de oráculo aleatorio [3], que está basado en la imposibilidad verificador de predecir la salida de la función hash. A continuación se presenta el protocolo de identificación de Schnorr, al que se le puede aplicar la heurística para utilizarlo en modo no interactivo.

A.2.3. Protocolo de identificación de Schnorr

Este protocolo se utiliza para demostrar el conocimiento del secreto del cifrado de ElGamal sin revelarlo [23]. Sus parámetros son: Dos números primos, p y q tales que se cumpla $q|p-1$ y un elemento g de orden q del grupo Z_p^*, es decir, debe cumplirse $g^q \equiv 1 \pmod p$. Al momento de utilizarse, el agente A es poseedor de una clave privada s, con la que calcula su clave pública $v \equiv g^s \pmod p$. El protocolo es el que sigue:

1.– El agente A genera aleatoriamente $r = 1 \dots q$, y a continuación le envía al verificador V el valor $x \equiv g^r \pmod p$.

2.– V guarda x y le devuelve a A un valor aleatorio e.

3.– A recibe e, con el que calcula $y \equiv r + se \pmod p$ y se lo manda a V.

4.– Una vez le llega y a V, este acepta si se cumple que $x \equiv g^y v^{-e} \pmod p$

Tras t repeticiones, la probabilidad de que el agente A esté mintiendo es de $\frac{1}{2^t}$. Además, se puede transformar a una versión no interactiva que puede utilizarse como firma [25]. Su generación de claves consiste en elegir un valor $a = 1 \dots q - 1$, que funciona como clave privada, generar $y \equiv g^a \pmod p$ y usar la tupla (p, q, g, y) como clave pública. La firma de mensajes se realiza mediante los siguientes pasos:

1.– Elegir secreto $k = 1 \dots q - 1$.

2.– Generar $r \equiv g^k \pmod{p}$.

3.– Generar $e = h(M||r)$.

4.– Generar $s \equiv ae + k \pmod{p}$.

5.– Se utiliza como firma la tupla (r, s)

Y para verificar que la tupla (r, s) es la firma de M, primero se calcula $e = h(M||r)$, y se comprueba si se cumple $r \equiv (g^s v^{-e} \pmod{p}$


UAM
UNIVERSIDAD AUTONOMA
DE MADRID